

ACL-DEMO 2023

**The 61st Annual Meeting of the Association for
Computational Linguistics: System Demonstrations**

Proceedings of the System Demonstrations

July 10-12, 2023

©2023 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 978-1-959429-70-8

Introduction

Welcome to the proceedings of the system demonstration track of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023) on July 9th – July 14th, 2023. For the ACL 2023 system demonstration track, we received a record number of 155 submissions, of which 58 were selected for inclusion in the program (acceptance rate of 37%) after being reviewed by at least three members of the program committee, while a small number of papers received only two reviews. We would like to thank the members of the program committee for their timely help in reviewing the submissions. Lastly, we thank the many authors that submitted their work to the demonstrations track. This year, the ACL conference is a hybrid event. The demonstration paper will be presented through pre-recorded talks and in presence during the poster sessions.

Danushka Bollegala, Ruihong Huang and Alan Ritter

ACL 2023 System Demonstration Chairs

Program Committee

Chairs

Danushka Bollegala, University of Liverpool/Amazon
Ruihong Huang, Texas A&M University
Alan Ritter, Georgia Institute of Technology

Program Committee

Micheal Abaho, University of Liverpool
Ahmed Abdelali, Qatar Computing Research Institute
Rodrigo Agerri, HiTZ Center - Ixa, University of the Basque Country UPV/EHU
Željko Agić, Unity Technologies
Alan Akbik, Humboldt-Universität zu Berlin
Zeynep Akkalyoncu, University of Waterloo
Khalid Al Khatib, Groningen University
Miguel A. Alonso, Universidade da Coruña
Rafael Anchiêta, Federal Institute of Piauı
Jakob Smedegaard Andersen, HAW Hamburg
John Arevalo, Universidad Nacional de Colombia
Eleftherios Avramidis, German Research Center for Artificial Intelligence (DFKI)
Ioana Baldini, IBM Research
Francesco Barbieri, Snap Inc.
Mohaddeseh Bastan, Stony Brook University
Khuyagbaatar Batsuren, National University of Mongolia
Timo Baumann, Ostbayerische Technische Hochschule Regensburg
Gábor Bella, University of Trento
Gábor Berend, University Of Szeged
Tyler Bikaun, The University of Western Australia
Yonatan Bisk, Carnegie Mellon University
Georgeta Bordea, Université de Bordeaux
Ari Bornstein, Bar Ilan University, Microsoft Corp
Aljoscha Burchardt, DFKI
Jan Buys, University of Cape Town
Ed Cannon, Expedia Group
Yixin Cao, Singapore Management University
Jiarun Cao, University of Manchester
Alberto Cetoli, Private
Yee Seng Chan, Raytheon BBN Technologies
Sachin Chanchani, Texas A&M University
Wanxiang Che, Harbin Institute of Technology
Pei Chen, Texas A&M University
Chung-chi Chen, National Institute of Advanced Industrial Science and Technology
Lu Chen, Shanghai Jiao Tong University
Guanyi Chen, Utrecht University
Yulin Chen, Tsinghua University
Jhih-jie Chen, National Tsing Hua University
Hai Leong Chieu, DSO National Laboratories
Yagmur Gizem Cinar, Amazon

Simone Conia, Sapienza University of Rome
Danilo Croce, University of Roma, Tor Vergata
Yiming Cui, Joint Laboratory of HIT and iFLYTEK Research
Shaobo Cui, Alibaba Group
Marina Danilevsky, IBM Research
Alok Debnath, Trinity College, Dublin
Jean-benoit Delbrouck, Stanford University
Shumin Deng, National University of Singapore
Yuntian Deng, Harvard University
Michael Desmond, IBM Research
Joseph P. Dexter, Harvard University
Victor Dibia, Microsoft Research
Chenchen Ding, NICT
Liam Dugan, University of Pennsylvania
Carl Edwards, University of Illinois, Urbana-Champaign
Carlos Escolano, Universitat Politècnica de Catalunya
Luis Espinosa Anke, Cardiff University
Kshitij Fadnis, IBM Research
Paulo Fernandes, Merrimack College
Anthony Ferritto, IBM
Dimitris Galanis, Institute for Language and Speech Processing, Athena Research Center
Sudeep Gandhe, Google Inc
Revanth Gangi Reddy, University of Illinois, Urbana Champaign
Xiang Gao, Microsoft Research
Aayush Gautam, Texas A&M University
Mozhdeh Gheini, University of Southern California
Stefan Grünewald, University of Stuttgart
Shachi H Kumar, Intel Labs
Huda Hakami, Taif University
Chi Han, University of Illinois at Urbana-Champaign
Xu Han, Tsinghua University
Xianpei Han, Institute of Software, Chinese Academy of Sciences
Yun He, Meta
Han He, Emory University
Ales Horak, Masaryk University
Dirk Hovy, Bocconi University
Xiaodan Hu, University of Illinois at Urbana-Champaign
Xuanjing Huang, Fudan University
Hen-hsen Huang, Institute of Information Science, Academia Sinica
Ali Hürriyetoglu, KNAW
Shajith Ikbal, IBM Research AI, India.
Ayyoob Imanigooghari, LMU Munich
Takumi Ito, Tohoku University / Langsmith Inc. / Utrecht University
Jeff Jacobs, Columbia University
Sai Muralidhar Jayanthi, AWS AI Labs
Feng Ji, Tencent Technology Ltd.
Zhuoxuan Jiang, Tencent
Haiyun Jiang, Tencent AI Lab
Ridong Jiang, Institute for Infocomm Research
Zhuoran Jin, Institute of Automation, Chinese Academy of Sciences
Sudipta Kar, Amazon Alexa AI

Eugene Kharitonov, Facebook AI
Philipp Koehn, Johns Hopkins University
Mamoru Komachi, Hitotsubashi University
Valia Kordoni, Humboldt-Universität zu Berlin
Lun-wei Ku, Academia Sinica
Harshit Kumar, IBM Research
Vishwajeet Kumar, IBM Research AI
Philippe Laban, Salesforce Research
Tuan Lai, University of Illinois at Urbana-Champaign
Mark Last, Ben-Gurion University of the Negev
John Lee, City University of Hong Kong
Dong-ho Lee, University of Southern California
Sangkeun Lee, Korea University
Yuanyuan Lei, Texas A&M University
Yanran Li, The Hong Kong Polytechnic University
Manling Li, UIUC
Raymond Li, University of British Columbia
Jing Li, Department of Computing, The Hong Kong Polytechnic University
Xintong Li, Apple
Sha Li, University of Illinois Urbana-Champaign
Sai Ramana Reddy Lingam, Texas A&M University
Marina Litvak, Shmoon College of Engineering
Wei Liu, The University of Western Australia
Lemao Liu, Tencent AI Lab
Changsong Liu, University of California, Los Angeles
Zhengzhong Liu, Carnegie Mellon University; Petuum INC.
Qian Liu, Sea AI Lab
Nikola Ljubešić, Jožef Stefan Institute
Daniel Loureiro, Cardiff University
Wei Lu, Singapore University of Technology and Design
Yubo Ma, Nanyang Technological University
Wolfgang Maier, Mercedes-Benz AG
Ramesh Manuvinakurike, Intel labs
Huisheng Mao, Tsinghua University
Alex Marin, Microsoft Corporation
Stella Markantonatou, ILSP/R.C. Athena"
Jonathan May, USC Information Sciences Institute
Ivan Vladimir Meza Ruiz, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas,
Universidad Nacional Autónoma de México
Md Messal Monem Miah, Texas A&M University
Margot Mieskes, University of Applied Sciences, Darmstadt
Gosse Minnema, University of Groningen
Yusuke Miyao, University of Tokyo
John Morris, Cornell Tech
Hamdy Mubarak, Qatar Computing Research Institute
Aldrian Obaja Muis, None
Philippe Muller, IRIT, University of Toulouse
Diane Napolitano, The Associated Press
Denis Newman-griffis, University of Pittsburgh
Tuan-phong Nguyen, Max Planck Institute for Informatics
Ansong Ni, Yale University

Andreas Niekler, Leipzig University
Tae-gil Noh, OMQ GmbH
Pierre Nugues, Lund University
Yusuke Oda, Inspired Cognition
Tsuyoshi Okita, Kyushu institute of technology
Takeshi Onishi, Toyota Motor Corporation
Daniel Ortega, University of Stuttgart
Xiaoman Pan, Tencent AI Lab
Alexandros Papangelis, Amazon Alexa AI
Xutan Peng, Huawei
Oren Pereg, Emergent AI Lab, Intel Labs
Stelios Piperidis, Athena RC/ILSP
Massimo Poesio, Queen Mary University of London
Prokopis Prokopidis, ILSP/Athena RC
Stephen Pulman, Apple Inc.
Ayesha Qamar, Texas A&M University
Carlos Ramisch, Aix Marseille University, CNRS, LIS
Ricardo Rei, Unbabel/INESC-ID
German Rigau, UPV/EHU
Andreas Rücklé, Amazon
Tulika Saha, University of Liverpool
Saurav Sahay, Intel Labs
Sashank Santhanam, University of North Carolina at Charlotte/ Apple
Sven Schmeier, Researcher
Fabian Schmidt, University of Wuerzburg
Procheta Sen, University of Liverpool
Sanuj Sharma, Google
Jiaming Shen, Google Research
Liang-hsin Shen, National Taiwan University
Michal Shmueli-scheuer, IBM Research
Lei Shu, Google Research
Sunayana Sitaram, Microsoft Research India
Amy Siu, Berliner Hochschule für Technik
Konstantinos Skianis, BLUAI
Mohammad Golam Sohrab, National Institute of Advanced Industrial Science and Technology
Yuanfeng Song, Hong Kong University of Science and Technology, WeBank Co., Ltd
Linfeng Song, Tencent AI Lab
Josef Steinberger, University of West Bohemia
Michael Stewart, The University of Western Australia
Carl Strathearn, Edinburgh Napier University
Jian Sun, China Mobile Research Institute
Chenkai Sun, University of Illinois at Urbana-Champaign
Jonathan Tong, Texas A&M University
Alexandra Uma, Queen Mary University of London
Lindsey Vanderlyn, University of Stuttgart
Natalia Vanetik, Shamoan College of Engineering
Andrea Varga, Theta Lake
Alakananda Vempala, Bloomberg
Dirk Väh, University of Stuttgart
Marilyn Walker, University of California Santa Cruz
Ziqi Wang, University of Illinois Urbana-Champaign

Jingjing Wang, Soochow University
Xuan Wang, Virginia Tech
Zijie J. Wang, Georgia Tech
Rui Wang, Vipshop (China) Co., Ltd.
Zhongqing Wang, Soochow University
Qingyun Wang, University of Illinois at Urbana-Champaign
Moritz Wolf, DFKI GMBH
Alina Wróblewska, Institute of Computer Science, Polish Academy of Sciences
Xianchao Wu, NVIDIA
Chien-sheng Wu, Salesforce
Deyi Xiong, Tianjin University
Qiongfai Xu, The University of Melbourne
Yujie Yang, tsinghua.edu.cn
Ziqing Yang, iFLYTEK Research
Tae Yano, Expedia Group
Wenlin Yao, Tencent AI Lab
Seid Muhie Yimam, Universität Hamburg
Pengfei Yu, Department of Computer Science, University of Illinois at Urbana-Champaign
Dian Yu, Google
Wenhao Yu, University of Notre Dame
Ziqi Yuan, Tsinghua University
Qi Zeng, University of Illinois at Urbana-Champaign
Qi Zhang, Fudan University
Chengzhi Zhang, Nanjing University of Science and Technology
Zixuan Zhang, University of Illinois Urbana-Champaign
Ningyu Zhang, Zhejiang University
Tianhui Zhang, University of Liverpool
Rongsheng Zhang, Netease Fuxi AI Lab
Jun Zhao, Chinese Academy of Sciences
Tiancheng Zhao, Binjiang Institute of Zhejiang University
Yi Zhou, Cardiff University
Cangqi Zhou, Nanjing University of Science and Technology
Guangyou Zhou, School of Computer Science, Central China Normal University
Imed Zitouni, Google
Erion Çano, Research Group Data Mining, University of Vienna
Gözde Şahin, Koç University

Table of Contents

Human-in-the-loop Schema Induction

Tianyi Zhang, Isaac Tham, Zhaoyi Hou, Jiakuan Ren, Leon Zhou, Hainiu Xu, Li Zhang, Lara Martin, Rotem Dror, Sha Li, Heng Ji, Martha Palmer, Susan Windisch Brown, Reece Suchocki and Chris Callison-Burch 1

PersLEARN: Research Training through the Lens of Perspective Cultivation

Yu-Zhe Shi, Shiqian Li, Xinyi Niu, Qiao Xu, Jiawen Liu, Yifan Xu, Shiyu Gu, Bingru He, Xinyang Li, Xinyu Zhao, Zijian Zhao, Yidong Lyu, Zhen Li, Sijia Liu, Lin Qiu, Jinhao Ji, Lecheng Ruan, Yuxi Ma, Wenjuan Han and Yixin Zhu 11

LAVIS: A One-stop Library for Language-Vision Intelligence

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, silvio savarese and Steven C.H. Hoi 31

Finspector: A Human-Centered Visual Inspection Tool for Exploring and Comparing Biases among Foundation Models

Bum Chul Kwon and Nandana Mihindukulasooriya 42

PrimeQA: The Prime Repository for State-of-the-Art Multilingual Question Answering Research and Development

Avi Sil, Jaydeep Sen, Bhavani Iyer, Martin Franz, Kshitij Fadnis, Mihaela Bornea, Sara Rosenthal, Scott McCarley, Rong Zhang, Vishwajeet Kumar, Yulong Li, Md Arafat Sultan, Riyaz Bhat, Juergen Bross, Radu Florian and Salim Roukos 51

Lingxi: A Diversity-aware Chinese Modern Poetry Generation System

Xinran Zhang, Maosong Sun, Jiafeng Liu and Xiaobing Li 63

Autodive: An Integrated Onsite Scientific Literature Annotation Tool

Yi Du, Ludi Wang, Mengyi Huang, Dongze Song, Wenjuan Cui and Yuanchun Zhou 76

A Practical Toolkit for Multilingual Question and Answer Generation

Asahi Ushio, Fernando Alva-Manchego and Jose Camacho-Collados 86

OpenSLU: A Unified, Modularized, and Extensible Toolkit for Spoken Language Understanding

Libo Qin, Qiguang Chen, Xiao Xu, Yunlong Feng and Wanxiang Che 95

SanskritShala: A Neural Sanskrit NLP Toolkit with Web-Based Interface for Pedagogical and Annotation Purposes

Jivnesh Sandhan, Anshul Agarwal, Laxmidhar Behera, Tushar Sandhan and Pawan Goyal . . . 103

LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models

Victor Dibia 113

MetaPro Online: A Computational Metaphor Processing Online System

Rui Mao, Xiao Li, Kai He, Mengshi Ge and Erik Cambria 127

DIAGRAPH: An Open-Source Graphic Interface for Dialog Flow Design

Dirk V  th, Lindsey Vanderlyn and Ngoc Thang Vu 136

disco: a toolkit for Distributional Control of Generative Models

Germ  n Kruszewski, Jos Rozen and Marc Dymetman 144

A Hyperparameter Optimization Toolkit for Neural Machine Translation Research

Xuan Zhang, Kevin Duh and Paul McNamee 161

<i>Japanese-to-English Simultaneous Dubbing Prototype</i>	
Xiaolin Wang, Masao Utiyama and Eiichiro Sumita	169
<i>VisKoP: Visual Knowledge oriented Programming for Interactive Knowledge Base Question Answering</i>	
Zijun Yao, YUANYONG CHEN, Xin Lv, Shulin Cao, Amy Xin, Jifan Yu, Hailong Jin, Jianjun Xu, Peng Zhang, Lei Hou and Juanzi Li	179
<i>PEEP-Talk: A Situational Dialogue-based Chatbot for English Education</i>	
Seunjun Lee, Yoonna Jang, Chanjun Park, Jungseob Lee, Jaehyung Seo, Hyeonseok Moon, Sugyeong Eo, Seounghoon Lee, Bernardo Yahya and Heuseok Lim	190
<i>OpenTIPE: An Open-source Translation Framework for Interactive Post-Editing Research</i>	
Fabian Landwehr, Thomas Steinmann and Laura Mascarell	208
<i>TencentPretrain: A Scalable and Flexible Toolkit for Pre-training Models of Different Modalities</i>	
Zhe Zhao, Yudong Li, Cheng Hou, Jing Zhao, Rong Tian, Weijie Liu, Yiren Chen, Ningyuan Sun, Haoyan Liu, Weiquan Mao, Han Guo, Weigang Gou, Taiqiang Wu, Tao Zhu, Wenhang Shi, Chen Chen, Shan Huang, Sihong Chen, Liqun Liu, Feifei Li, Xiaoshuai Chen, Xingwu Sun, Zhanhui Kang, Xiaoyong Du, Linlin Shen and Kimmo Yan	217
<i>NeuroX Library for Neuron Analysis of Deep NLP Models</i>	
Fahim Dalvi, Hassan Sajjad and Nadir Durrani	226
<i>SciLit: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation</i>	
Nianlong Gu and Richard H.R. Hahnloser	235
<i>Massively Multi-Lingual Event Understanding: Extraction, Visualization, and Search</i>	
Chris Jenkins, Shantanu Agarwal, Joel Barry, Steven Fincke and Elizabeth Boschee	247
<i>YANMTT: Yet Another Neural Machine Translation Toolkit</i>	
Raj Dabre, Diptesh Kanojia, Chinmay Sawant and Eiichiro Sumita	257
<i>XMD: An End-to-End Framework for Interactive Explanation-Based Debugging of NLP Models</i>	
Dong-Ho Lee, Akshen Kadakia, Brihi Joshi, Aaron Chan, Ziyi Liu, Kiran Narahari, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara and Xiang Ren	264
<i>OpenDelta: A Plug-and-play Library for Parameter-efficient Adaptation of Pre-trained Models</i>	
Shengding Hu, Ning Ding, Weilin Zhao, Xingtai Lv, Zhen Zhang, Zhiyuan Liu and Maosong Sun	274
<i>Hierarchy Builder: Organizing Textual Spans into a Hierarchy to Facilitate Navigation</i>	
Itay Yair, Hillel Taub-Tabib and Yoav Goldberg	282
<i>CARE: Collaborative AI-Assisted Reading Environment</i>	
Dennis Zyska, Nils Dycke, Jan Buchmann, Ilia Kuznetsov and Iryna Gurevych	291
<i>The ROOTS Search Tool: Data Transparency for LLMs</i>	
Aleksandra Piktus, Christopher Akiki, Paulo Villegas, Hugo Laurençon, Gérard Dupont, Sasha Luccioni, Yacine Jernite and Anna Rogers	304
<i>The OPUS-MT Dashboard – A Toolkit for a Systematic Evaluation of Open Machine Translation Models</i>	
Jörg Tiedemann and Ona de Gibert	315
<i>The D-WISE Tool Suite: Multi-Modal Machine-Learning-Powered Tools Supporting and Enhancing Digital Discourse Analysis</i>	
Florian Schneider, Tim Fischer, Fynn Petersen-Frey, Isabel Eiser, Gertraud Koch and Chris Bie-mann	328

<i>OpenRT: An Open-source Framework for Reasoning Over Tabular Data</i> Yilun Zhao, Boyu Mi, Zhenting Qi, Linyong Nan, Minghao Guo, Arman Cohan and Dragomir Radev	336
<i>UINAUIL: A Unified Benchmark for Italian Natural Language Understanding</i> Valerio Basile, Livio Bioglio, Alessio Bosca, Cristina Bosco and Viviana Patti	348
<i>Zshot: An Open-source Framework for Zero-Shot Named Entity Recognition and Relation Extraction</i> Gabriele Picco, Marcos Martinez Galindo, Alberto Purpura, Leopold Fuchs, Vanessa Lopez and Thanh Lam Hoang	357
<i>BiSync: A Bilingual Editor for Synchronized Monolingual Texts</i> Josep Crego, Jitao Xu and François Yvon	369
<i>Riveter: Measuring Power and Social Dynamics Between Entities</i> Maria Antoniak, Anjalie Field, Jimin Mun, Melanie Walsh, Lauren Klein and Maarten Sap ..	377
<i>Fast Whitespace Correction with Encoder-Only Transformers</i> Hannah Bast, Matthias Hertel and Sebastian Walter	389
<i>ESPnet-ST-v2: Multipurpose Spoken Language Translation Toolkit</i> Brian Yan, Jiatong Shi, Yun Tang, Hirofumi Inaguma, Yifan Peng, Siddharth Dalmia, Peter Polak, Patrick Fernandes, Dan Berrebbi, Tomoki Hayashi, Xiaohui Zhang, Zhaoheng Ni, Moto Hira, Soumi Maiti, Juan Pino and Shinji Watanabe	400
<i>CB2: Collaborative Natural Language Interaction Research Platform</i> Jacob Sharf, Mustafa Omer Gul and Yoav Artzi	412
<i>Inseq: An Interpretability Toolkit for Sequence Generation Models</i> Gabriele Sarti, Nils Feldhus, Ludwig Sickert and Oskar van der Wal	421
<i>Pipeline for modeling causal beliefs from natural language</i> John Priniski, Ishaan Verma and Fred Morstatter	436
<i>TabGenie: A Toolkit for Table-to-Text Generation</i> Zdeněk Kasner, Ekaterina Garanina, Ondrej Platek and Ondrej Dusek	444
<i>An Efficient Conversational Smart Compose System</i> Yun Zhu, Xiayu Chen, Lei Shu, Bowen Tan, Xinying Song, Lijuan Liu, Maria Wang, Jindong Chen and Ning Ruan	456
<i>Which Spurious Correlations Impact Reasoning in NLI Models? A Visual Interactive Diagnosis through Data-Constrained Counterfactuals</i> Robin Chan, Afra Amini and Mennatallah El-Assady	463
<i>LaTeX2Solver: a Hierarchical Semantic Parsing of LaTeX Document into Code for an Assistive Optimization Modeling Application</i> Rindra Ramamonjison, Timothy Yu, Linzi Xing, Mahdi Mostajabdaveh, Xiaorui Li, Xiaojin Fu, Xiongwei Han, Yuanzhe Chen, Ren Li, Kun Mao and Yong Zhang	471
<i>Alfred: A System for Prompted Weak Supervision</i> Peilin Yu and Stephen Bach	479
<i>OpenICL: An Open-Source Framework for In-context Learning</i> Zhenyu Wu, Yaoxiang Wang, Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Jingjing Xu and Yu Qiao	489

<i>Self-Supervised Sentence Polishing by Adding Engaging Modifiers</i>	
Zhexin Zhang, Jian Guan, Xin Cui, Yu Ran, Bo Liu and Minlie Huang	499
<i>Effidit: An Assistant for Improving Writing Efficiency</i>	
Shuming Shi, Enbo Zhao, Wei Bi, Deng Cai, Leyang Cui, Xinting Huang, Haiyun Jiang, Duyu Tang, Kaiqiang Song, Longyue Wang, Chenyan Huang, Guoping Huang, Yan Wang and Piji Li ...	508
<i>WizMap: Scalable Interactive Visualization for Exploring Large Machine Learning Embeddings</i>	
Zijie J. Wang, Fred Hohman and Duen Horng Chau	516
<i>A System for Answering Simple Questions in Multiple Languages</i>	
Anton Razzhigaev, Mikhail Salnikov, Valentin Malykh, Pavel Braslavski and Alexander Panchenko	524
<i>KWJA: A Unified Japanese Analyzer Based on Foundation Models</i>	
Nobuhiro Ueda, Kazumasa Omura, Takashi Kodama, Hirokazu Kiyomaru, Yugo Murawaki, Daisuke Kawahara and Sadao Kurohashi	538
<i>Disease Network Constructor: a Pathway Extraction and Visualization</i>	
Mohammad Golam Sohrab, Khoa Duong, Goran Topić, Masami Ikeda, Nozomi Nagano, Yayoi Natsume-Kitatani, Masakata Kuroda, Mari Itoh and Hiroya Takamura	549
<i>Petals: Collaborative Inference and Fine-tuning of Large Models</i>	
Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Maksim Riabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin and Colin Raffel	558
<i>UKP-SQuARE v3: A Platform for Multi-Agent QA Research</i>	
Haritz Puerto, Tim Baumgärtner, Rachneet Sachdeva, Haishuo Fang, Hao Zhang, Sewin Tariverdian, Kexin Wang and Iryna Gurevych	569
<i>Ranger: A Toolkit for Effect-Size Based Multi-Task Evaluation</i>	
Mete Sertkan, Sophia Althammer and Sebastian Hofstätter	581
<i>GAIA Search: Hugging Face and Pyserini Interoperability for NLP Training Data Exploration</i>	
Aleksandra Piktus, Odunayo Ogundepo, Christopher Akiki, Akintunde Oladipo, Xinyu Zhang, Hailey Schoelkopf, Stella Biderman, Martin Potthast and Jimmy Lin	588
<i>DeepPavlov Dream: Platform for Building Generative AI Assistants</i>	
Diliara Zharikova, Daniel Kornev, Fedor Ignatov, Maxim Talimanchuk, Dmitry Evseev, Ksenya Petukhova, Veronika Smilga, Dmitry Karpov, Yana Shishkina, Dmitry Kosenko and Mikhail Burtsev	599

Human-in-the-Loop Schema Induction

Tianyi Zhang^{*1}, Isaac Tham^{*1}, Zhaoyi Hou^{*1}, Jiaxuan Ren¹, Liyang Zhou¹
Hainiu Xu¹, Li Zhang¹, Lara J. Martin¹, Rotem Dror¹, Sha Li², Heng Ji²
Martha Palmer³, Susan Brown³, Reece Suchocki³, and Chris Callison-Burch¹

¹ University of Pennsylvania, ² University of Illinois Urbana-Champaign

³ University of Colorado, Boulder, * equal contribution

{zty, joeyhou, ccb}@upenn.edu

Abstract

Schema induction builds a graph representation explaining how events unfold in a scenario. Existing approaches have been based on information retrieval (IR) and information extraction (IE), often with limited human curation. We demonstrate a human-in-the-loop schema induction system powered by GPT-3.¹ We first describe the different modules of our system, including prompting to generate schematic elements, manual edit of those elements, and conversion of those into a schema graph. By qualitatively comparing our system to previous ones, we show that our system not only transfers to new domains more easily than previous approaches but also reduces efforts of human curation thanks to our interactive interface.

1 Introduction

Event-centric natural language understanding (NLU) has been increasingly popular in recent years. Systems built from an event-centric perspective have resulted in impressive improvements in numerous tasks, including open-domain question answering (Yang et al., 2003), intent prediction (Rashkin et al., 2018), timeline construction (Do et al., 2012), text summarization, (Daumé and Marcu, 2006) and misinformation detection (Fung et al., 2021). At the heart of event-centric NLU lie event schemas, an abstract representation of how complex events typically unfold. The study for such a representation dates back to the 70s, where scripts were proposed as a series of sequential actions (Roger C. Schank, 1977). Back then, the schemas were limited to linear and temporal ones. A more recent formulation of event schemas is a graph where the vertices are event flows and the edges are temporal or hierarchical relations between those events (Du et al., 2022).

¹Webpage: <https://www.kairos.jiaxuan.me>;
Video: <https://www.youtube.com/watch?v=myru-fozVWI>

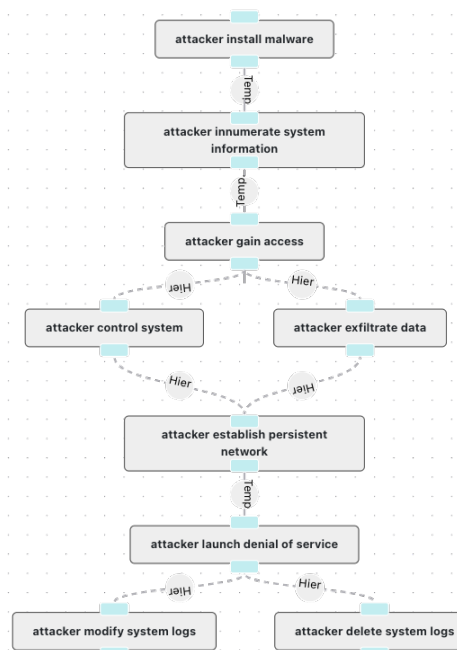


Figure 1: An example of Cyber Attack schema. The tree structure represents the temporal and hierarchical relations between the nodes.

For example, as shown in Figure 1, the event schema for a "cyber attack" could include sub-events such as "gain access", "control system", "exfiltrate files", "modify system logs", etc. The schema would also include the relationships between these sub-events. For instance, the event "gain access" would take place *before* the event "modify system logs" since a person needs access to a system before modifying it. For the same reason, "exfiltrate data" would only take place *after* "gain access". Event schemas like this encode high-level knowledge about the world and allow artificial intelligence systems to reason about unseen events (Du et al., 2022).

The DARPA Knowledge-directed Artificial Intelligence Reasoning Over Schemas (KAIROS) program² aims at developing schema-based AI sys-

²<https://www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-sch>

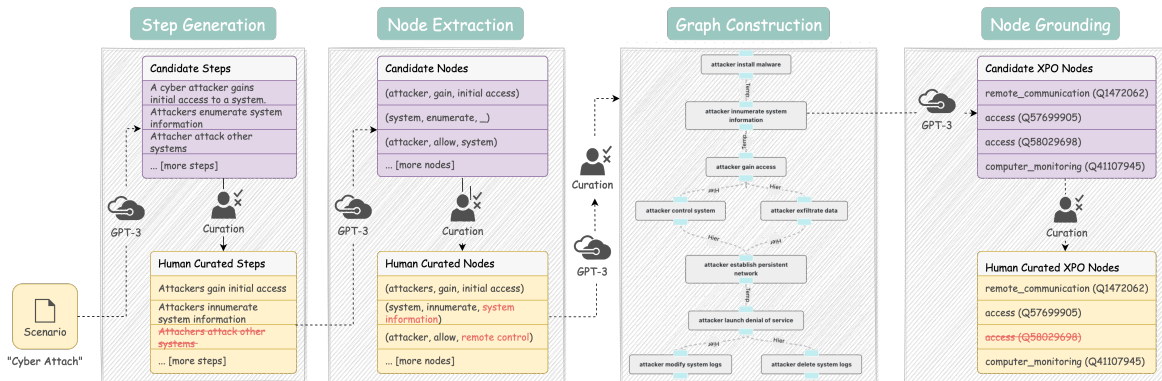


Figure 2: Our schema curation system includes four main stages: Step Generation, Node Extraction, Graph Construction and Node Grounding; Model output is highlighted in purple background; Human curated output is highlighted in yellow background; human curation is shown in red.

tems that can identify, comprehend, and forecast complex events in a diverse set of domains. To enable such a system, scalable generation of high-quality event schemas is very crucial. On one hand, fully-manual schema creation at a large scale can be inefficient, since people have diverse views about a certain concept, leading to inconsistent schema results. On the other hand, fully automated systems are scalable, but not with high-quality. In fact, the majority of existing approaches under the KAIROS program are fully-automated IR and IE systems over large collections of news articles (Li et al., 2020, 2021). Only some of limited human post-processing on schemas (Ciosici et al., 2021) have been explored. Further discussion of the advantages and limitations of existing systems can be found in Related Work.

Instead of focusing on fully-automated schema induction systems, we propose a human-in-the-loop schema induction pipeline system. Rather than using IR and IE over a large document collection, our system relies on pre-trained large language models (LLMs) and human intervention to jointly produce schemas. Our main motivation is that human-verified schemas are of higher quality. That is because human curation can filter out failure cases such as incompleteness, instability, or poor domain transfer results in previous systems (Dror et al., 2022; Peng et al., 2019). With human curation, schemas are more reliable and accountable when applied to downstream tasks such as event prediction. This is significant if the downstream tasks involve safety-critical applications like epidemic prevention, where the quality of the schema matters beyond task performance numbers.

Figure 2 is a flowchart of our four-stage schema induction system: **step generation, node extraction, graph construction, and node grounding**. Each stage has two main components: the LLM (e.g. GPT-3) at the back-end to output predictions (the purple boxes in the figure) and an interactive interface at the front-end for human curation of the model output (the yellow boxes). The GPT-3 prompts that are used in each stage of the process are given in the Appendix A, along with example inputs and outputs.

A more comprehensive description of the implementation and functionalities of our interface can be found in Section 4. A case study is given in Section 5. It walks through each step in our pipeline system under an example scenario, cyber attack. Also, in Section 5, we provide a qualitative evaluation of five example scenarios. The summary and discussion of our system are included in Section 6.

2 Related Work

2.1 Schema Induction

Early work from Chambers and Jurafsky (2008, 2009) automatically learned a schema from newswire text based on coreference and statistical probability models. Later, Peng and Roth (2016); Peng et al. (2019) generated an event schema based on their proposed semantic language model (like an RNN structure). Their work represented the whole schema as a linear sequence of abstract verb senses like `arrest . 01` from VerbNet (Schuler, 2005). Those works had two main shortcomings: first, the schema was created for a single actor (protagonist), e.g. suspect. It caused limited coverage in a more complex scenario, e.g. business change-acquisition; second, the generated schema, a simple

linear sequence, failed to consider different alternatives such as XOR.

More recently, Li et al. (2020, 2021) used transformers to handle schema generation in a complex scenario. It viewed a schema as a graph instead of a linear sequence. However, this approach was unable to transfer to new domains where the supervised event retrieval and extraction model failed. Dror et al. (2022) took GPT-3 generated documents to build a schema. Although it bypassed the event retrieval and extraction process and solved the domain transfer problem, it suffered from the incompleteness and instability of GPT-3 outputs.

Currently, neither do they offer a perfect solution for schema induction without manual post-processing, nor build a timely human correction system (Du et al., 2022). Our demonstration system develops a curation interface that can generate a comprehensive schema easily with a human curator in the loop. The curated data collected through our tool could be useful for fine-tuning and improving the models.

2.2 Human-in-the-loop Schema Curation Interface

Another area related to our work is human-in-the-loop schema generation, where annotators collaborate with computational models to create high-quality event schema. In this field, one of the closest approaches is the Machine-Assisted Script Curation (Ciosici et al., 2021) created for script induction. With a fully interactive interface, they have shown the feasibility of realtime interaction between humans and pre-trained LLMs (e.g. GPT-2 or GPT-3). The main differences are the level of automation and adaptability to other generative models. In terms of automation, our interface makes use of pre-trained LLMs to automatically generate schema content, compared to their interface which largely counts on human input. For adaptability, our interface supports the curation of the schema generated by different language models (e.g. GPT-3 models with different sizes), which makes it possible for users to evaluate the generations of different models. In contrast, there is no such possibility in their interface.

Another interface built for schema curation focuses on visualization of the schema structure, such as the temporal relations between event nodes and internal relations among entities (Mishra et al., 2021). While this interface provides a user-friendly

experience when it comes to schema graph curation, it requires the user to come up with the content of event schemas in json format, which requires much more human effort compared to our interface. In addition, our interface also provides an optional grounding function after the event graph curation step, which is not presented in this interface.

3 Terminology and Problem Definition

Our work focuses on efficiently building a schema graph of a scenario using both LLMs and human input. Following the workflow of our system (see the workflow in Figure 2), a **scenario** is a general event type that an interested party will build the schema for, e.g. a ‘disease outbreak’. **Steps** are a list of sub-events generated by GPT-3 according to a prompt in the step generation stage. Each step can be a phrase or a short sentence, such as ‘spread to other areas’, etc. **Nodes** or **tuples** are subject-verb-object pairs extracted from steps at the node extraction stage, such as ‘(disease, spread, to other area)’. **Graphs** are a visualization of the schema, whose edges joining the nodes represent temporal and hierarchical relations.

4 Implementation

Our pipeline system contains four sequential stages: **step generation**, **node extraction**, **graph construction**, and **node grounding**. A flowchart of the interface system is shown in Figure 2. The **step generation** stage generates steps for a scenario and the user can specify how many steps they would like to generate. The **node extraction** stage extracts nodes (subject-verb-object tuples) from the previous verbose steps. The **graph construction** stage orders the extracted nodes temporally and hierarchically. Meanwhile, modifications of the nodes are still possible. The **node grounding** stage maps node text to a node in the XPO ontology (Elizabeth Spaulding et al., In preparation) (derived from WikiData³). The flexible interface system allows users to either go through the entire process to create a schema from scratch or directly start at any stage to edit the model’s prediction. In addition, the back-end GPT-3 models can be replaced by other user pre-trained models if deployed locally.

³https://www.wikidata.org/wiki/Wikidata:Main_Page

4.1 Step Generation

The step generation stage aims at generating steps given a scenario. At the backend, zero-shot GPT-3 incorporates a user’s input into a prompt and generates ordered steps. The interface allows users to generate steps quickly with prompt templates⁴ or finetune the generated steps with user-designed prompts. A typical use case of the user-designed prompts is to expand a certain step to more detailed steps. For instance, a template prompt "List the steps involved in {disease outbreak}:" may create steps such as "1. Identify the symptoms of the disease; 2. Collect data from affected individuals; ...". Then, the user can re-prompt for, e.g., the second step, "List the steps involved {step2} in detail:". Additionally, users can modify and select GPT-3 generated steps easily by clicking on them. When the 'save' button is clicked, all user selected steps will be saved in the database for the use of the node extraction stage or further fine-tuning of the step generation model. A screenshot of the step generation interface with user’s operations can be seen in Figure 3.

4.2 Node Extraction

Nodes are structured representations of events in the form of a {subject, verb, object} tuple. Node extraction is to extract these nodes from the GPT-3 generated steps saved in step generation stage, which are unstructured sentences.

There are two methods, based on AllenNLP (Shi and Lin, 2019) or GPT-3, that users can choose from to extract nodes. The former uses AllenNLP’s Semantic Role Labelling (SRL) model to extract nodes from the steps. The SRL model implements a BERT (Devlin et al., 2018) sequence prediction model to identify the predicates and the arguments (e.g. A0, A1) in a text. We simply choose the identified A0 as subject, A1 as object, and predicate as the verb to form a node. An optional coreference resolution model can be used to resolve referenced entities between the different steps with an AllenNLP’s SpanBERT-based model (Lee et al., 2018). Here, we concatenate all the steps and replace a pronoun with its referenced entity (noun) in the original steps.

The GPT-3 node extraction method uses instructional few-shot prompting to extract {subject, verb, object} tuples from the steps. Several example sen-

⁴an {event type} appended to a predefined prompt: Before, After or Sub-steps

tences are given to show GPT-3 the expected syntactic and semantic output. We follow (Liu et al., 2022)’s recommendation for few-shot design by including context examples that are semantically similar to the KAIROS application environment (daily life and news). See appendix A for our few-shot prompts.

The extracted nodes are shown to the user in a table with 3 columns (subject, verb, object). For example, for "The CDC collects and analyzes data on disease outbreaks", one of the extracted nodes is "The CDC (subject) collects (verb) data (object)". Users are able to choose and edit nodes (tuples). User edits are saved and will be used for graph construction and fine-tuning of the GPT-3 node extraction model.

4.3 Graph Construction

In the graph construction stage, our system automatically adds temporal and hierarchical edges to the previously extracted nodes. The edges are created using zero-shot GPT-3 with multiple choice questions. For each pair of nodes, GPT-3 is instructed to choose between 'Before', 'After', 'Same time' or 'no relation' for temporal edges; and 'Parent', 'Child' or 'no relation' for hierarchical edges. For example, for the node pair "collect data" and "identify the signs and symptoms", GPT-3 predicts 'After' for temporal order and 'no relation' for hierarchical order, in which case we will add a temporal edge from "identify the signs and symptoms" to "collect data", and no hierarchical edge will be created. If a conflict occurs between (node1, node2) pair and (node2, node1) pair, e.g. 'After' and 'After' for a temporal order or 'Parent' and 'Parent' for a hierarchical order, we will treat it as no relation to resolve the conflict, thus adding no new edges to the graph.

The graph construction interface allows users to modify the GPT-3 generated schema with ease. After predicting both temporal and hierarchical relations between all pairs of nodes, the interface will display the graph via the Vis-network framework⁵. It supports adding, editing, deleting graph nodes and edges. When the user clicks on a node, the detailed information including the ID and description of a node will be shown as well as the button to delete or edit the node. By clicking the edge, users can modify the edge type or delete it. Users will be

⁵<https://www.npmjs.com/package/react-vis-network-graph>

able to create a new node by double clicking and a new edge by dragging and dropping an arrow from two nodes. A screenshot of our graph construction interface can be seen in figure 4.

4.4 Node Grounding

Although a schema (graph) is completely created after the previous stages, some nodes may express the same semantic information, e.g., “refugees flee” and “refugees ran away”. To ensure the reliability and comparability of created schemas, our system grounds the nodes to an ontology, namely the XPO ontology, in the last stage. Each node in the XPO ontology contains a unique node ID, a node name, and a concise description (definition), and a list of similar nodes. Our system offers two ways of grounding, “name inference grounding” or “name similarity grounding”. Name inference grounding maps the schema nodes to XPO nodes by predicting the XPO node’s name; name similarity grounding finds the XPO nodes by comparing the similarities between the embeddings of a schema node and a XPO node’s name.

In name inference grounding, given a graph node, our system first uses few-shot GPT-3 to deduce a list of possible XPO names (see few-shot prompt example in appendix A). Then, the candidate XPO names are postprocessed by dropping off the wrong prediction and adding similar XPO names to the true prediction. After that, each possible XPO name will be checked for entailment with the original graph node. The entailment model is a BART-large model fine-tuned on the MNLI dataset (Lewis et al., 2020; Williams et al., 2018). The input is the original graph node as the premise and the possible XPO name as the hypothesis, and the output is the entailment score. We sort the possible XPO node names by their entailment scores. Users can view and choose from the top- k suggested XPO nodes for the grounding of the original graph node. In name similarity grounding, the top- k related XPO nodes are retrieved by computing the cosine-similarity of the GloVe embedding between the graph node and the name of XPO nodes (Pennington et al., 2014). The above two methods are complementary to each other especially when users cannot find expected XPO nodes with one method. Human-curated data is saved in the back-end database. A screenshot of node grounding can be seen in Figure 5.

5 Evaluation

5.1 A Case Study

In this section, we walk through the whole process of creating a toy schema with our interface which is much simpler than a fully developed schema. We assume the scenario is ‘cyber attack’.

In the step generation stage, users can form a prompt from templates such as “list the steps involved in a cyber attack” with ‘cyber attack’ as the name and sub-event as the prompt type. Then, GPT-3 will generate 5 steps. For example, “1. A cyber attacker gains initial access to a system” and “5. The attacker exfiltrates data from the compromised system.” Users can modify the content and choose steps to save. For example, one may change the first step to “1. A cyber attacker access a system.” and save the step. See a screenshot of five steps for reference in figure 3.

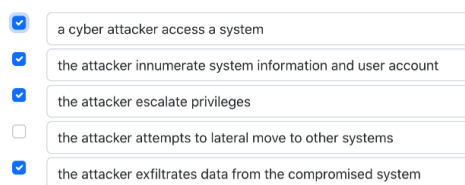


Figure 3: A sample of generated steps after human-curation for scenario ‘cyber attack’.

Next, in the node extraction stage, GPT-3 will be prompted to extract nodes from the selected steps. For example, GPT-3 will output {cyber attacker, access, system} for the first step. The user can change the outputs to correct any mistakes. In this sample, we extract 4 nodes, they are: {cyber attacker, access, system}, {attacker, enumerate, system information and user account}, {attacker, escalates, privileges}, {attacker, exfiltrate, data}. And we concatenate the {subject, verb, object} into a piece of text as a node for the next stage.

Thereafter, in the graph construction stage, we prompt GPT-3 to automatically build linear temporal edges on the above four nodes that users can modify. We manually add a scenario node ‘cyber attack’ and link with the other four existing nodes through hierarchical edges. see a screenshot of the graph in figure 4.

Finally, we can optionally ground our graph node into the XPO ontology. For example, the node “cyber attacker access system” can be mapped to choices of ‘access’, ‘computer monitoring’, ‘remote communicating’ using name similarity grounding. In this case, we don’t get any results

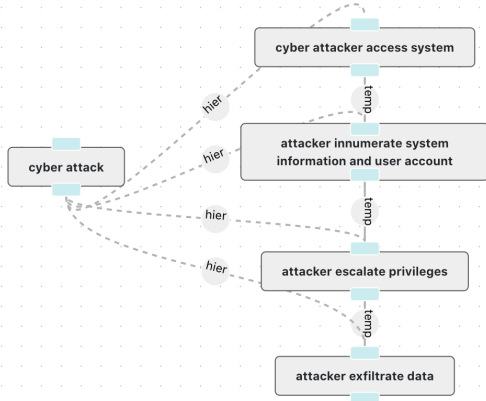


Figure 4: A sample of a constructed graph after human-curation for scenario 'cyber attack'

from name inference grounding. See a screenshot of grounding in Figure 5.

The name of event you wish to ground

Ground

- name** : remote_communication
similarity score : 0.81
wd_description : communication between two parties in remote locations, either concurrent or non-concurrent; correspondence and telecommunication
wd_node : Q1472062
- name** : access
similarity score : 0.82
wd_description : right, permission, or ability to approach, enter, or interact with something
wd_node : Q57699905
- name** : access
similarity score : 0.82
wd_description : set of mechanisms by which a target can be reached
wd_node : Q58029698
- name** : computer_monitoring
similarity score : 0.83
wd_description : None
wd_node : Q41107945

Figure 5: Top-4 XPO node choices of graph node "cyber attacker access system".

5.2 User Evaluation

We followed the evaluation methodology used by Ciosici et al. (2021) with slight modifications to assess our system. Evaluation is done by researchers in the field of NLP who have experience in hand-writing event schemas but have not used the interface before. In the step generation and node extraction stage, we count the number of human selected steps/nodes out of the total number of machine generated results as accuracy. For simplicity, we ignore users' modifications (e.g. rephrasing) at this point. In the graph construction stage, we compare how many nodes and edges are modified

	EVC	FOD	JOB	MED	MRG
Step Acc	11/12	7/8	10/10	10/10	12/12
Node Acc	13/15	10/10	11/12	12/12	12/14
Graph Node ED	1	0	0	0	0
Graph Edge ED	8	0	7	3	16
Grounding Success Rate	5/12	3/10	3/11	6/12	9/12
Self-reported time (min)	15	10	11	10	14

Table 1: User evaluation results. Acc in line 1 and 2 represents Accuracy. ED in line 3 and 4 means Editing Distance. Ciosici et al. (2021)'s approach, on average, took an hour to create the schema of a scenario.

(added or deleted) using graph edit distance. In the grounding part, the success rate is measured as successful retrieval of at least one relevant XPO node within top-3 grounding results for a given event node. We also ask users to self-report their total time of interaction. For all the evaluations, we use GPT-3 Davinci model as the language model.⁶

We follow prior work and evaluate our system on five scenarios: Evacuation (EVC), Ordering Food in a Restaurant (FOD), Finding and Starting a New Job (JOB), Obtaining Medical Treatment (MED), Corporate Merger or Acquisition (MRG).⁷

As shown in Table 1, our interactive system shows high accuracy in step and node generation phases, thanks to the richness of world knowledge from LLMs. However, the graph construction and the node grounding require more human curation, due to the difficulty of event reasoning, such as the understanding of temporal and hierarchical relationships; and the retrieval ability from large database. In those cases, we showed that human curation can step in timely and improve the quality of event schema when LLM-based models make mistakes⁷. In addition, our interface is easy to use, with much shorter time required to complete each event schema task compared to previous work (Ciosici et al., 2021).

We also report a qualitative study introducing the types of human modifications on the automated generations. At the step generation stage, GPTs aren't likely to make commonsense and grammar errors. However, if its required to generate more steps, it may be susceptible to redundancy, such as, "A does B" and "A finishes doing B", and hu-

⁶<https://platform.openai.com/docs/models/gpt-3>

⁷Detailed evaluation results: <https://joeyhou.notion.site/Human-in-the-Loop-Schema-Induction-Interface-Logs-1eb52403b05542919ccea214656f4211>

man removes these steps. Then, for the node extraction, results can be simplistic and ambiguous when the original sentence contains rich information, such as location, condition, or other modifiers. For example, given the step "waitress bring order to the kitchen", automatic node extraction produces "(waitress, bring, order)", while human needs to add back some necessary components, e.g. the location information "kitchen" or constraint "food order". Last, for the graph construction, current graph is often linear based on the previous nodes' order, human efforts play an essential role to elaborate on the specific relations including AND, OR. For example, "person updates the resume" and "person tailors the cover letter" are independent and can be concurrent, not sequential.

6 Conclusion

With the acknowledgements that fully depending on human annotation is expensive and inefficient, while wholly automated generations can be unreliable, we propose a human-in-the-loop schema curation interface with pre-trained large language models (LLMs) as the backbone. We use LLMs to generate candidate components of a schema and involve human as the final judge for both the content and structure of the event schema. With empirical evaluations, we show that our system can efficiently produce human-validated event schemas with minor human efforts.

Limitations

We have several limitations in our current approach. First, our current system uses zero-shot or few-shot to prompt GPT-3 without any fine tuning. In future work, we plan to fine-tune our GPT-3 with human curated data that we collect. We expect that fine-tuning will improve our models' performance. It may also be possible to use human curated data to train a policy network recommended by OpenAI (Ouyang et al., 2022). Second, we can replace GPT-3 with more robust task specific models at some stages, e.g., the pre-trained model for predicting temporal and hierarchical orders. Third, some users suggested incorporating a graph view at the other three stages, which will help users to generate based on the current graph. We will include this graph view in our next version. Forth, our current evaluation is experimental and probably subjective, we will develop more robust evaluation metrics comparing manual, Ciosici et al. (2021)'s and our

schema and test on downstream tasks in the next step.

Ethics Statement

To our knowledge, our back-end GPT-3 model was trained mainly on English web data, it may prefer events happen in an English environment. Furthermore, our test showed that it generated events specifically fit in American setting, for example, Miranda Rights for arrest, Democrats and Republicans in United States for election. These facts suggest GPT-3 may ignore the knowledge of non-American cultures or minority groups. In addition, currently, we only create schemas for scenarios that are reported in mainstream news media, e.g. conflict, communication. It excludes the schemas from other domains, such as biology, medicine.

7 Acknowledgements

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), the IARPA BETTER Program (contract 2019-19051600004 and 2019-19051600006), the IARPA HIATUS Program (contract 2022-22072200005), and the NSF (Award 1928631) and National Science Foundation under Grant #2030859 to the Computing Research Association for the CIFellows Project. Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, DARPA, IARPA, NSF, or the U.S. Government.

We thank researchers in PennNLP groups, and from other universities who gave us suggestions on the paper.

References

- Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.
- Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610.
- Manuel Ciosici, Joseph Cummings, Mitchell DeHaven, Alex Hedges, Yash Kankanampati, Dong-Ho Lee,

- Ralph Weischedel, and Marjorie Freedman. 2021. [Machine-assisted script curation](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 8–17, Online. Association for Computational Linguistics.
- Hal Daumé and Daniel Marcu. 2006. [Bayesian query-focused summarization](#). In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, page 305–312, USA. 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*.
- Quang Do, Wei Lu, and Dan Roth. 2012. [Joint inference for event timeline construction](#). In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687, Jeju Island, Korea. Association for Computational Linguistics.
- Rotem Dror, Haoyu Wang, and Dan Roth. 2022. Zero-shot on-the-fly event schema induction. *arXiv preprint arXiv:2210.06254*.
- Xinya Du, Zixuan Zhang, Sha Li, Pengfei Yu, Hongwei Wang, Tuan Lai, Xudong Lin, Ziqi Wang, Iris Liu, Ben Zhou, Haoyang Wen, Manling Li, Darryl Hannan, Jie Lei, Hyounghun Kim, Rotem Dror, Haoyu Wang, Michael Regan, Qi Zeng, Qing Lyu, Charles Yu, Carl Edwards, Xiaomeng Jin, Yizhu Jiao, Ghazaleh Kazeminejad, Zhenhailong Wang, Chris Callison-Burch, Mohit Bansal, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, Martha Palmer, and Heng Ji. 2022. [RESIN-11: Schema-guided event prediction for 11 newsworthy scenarios](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pages 54–63, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.
- Anatole Gershman Elizabeth Spaulding, Susan Brown Rosario Uceda-Sosa, Peter Anick James Pustejovsky, and Martha Palmer. In preparation. The darpa wiki-data overlay: Wikidata as an ontology for natural language processing.
- Yi Fung, Christopher Thomas, Revanth Gangi Reddy, Sandeep Polisetty, Heng Ji, Shih-Fu Chang, Kathleen McKeown, Mohit Bansal, and Avi Sil. 2021. [InfoSurgeon: Cross-media fine-grained information consistency checking for fake news detection](#). 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 1683–1698, Online. Association for Computational Linguistics.
- Kenton Lee, Luheng He, and L. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*.
- 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.
- Manling Li, Sha Li, Zhenhailong Wang, Lifu Huang, Kyunghyun Cho, Heng Ji, Jiawei Han, and Clare Voss. 2021. The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5203–5215.
- Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. [What makes good in-context examples for GPT-3?](#) In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.
- Piyush Mishra, Akanksha Malhotra, Susan Windisch Brown, Martha Palmer, and Ghazaleh Kazeminejad. 2021. A graphical interface for curating schemas. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 159–166.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.
- Haoruo Peng, Qiang Ning, and Dan Roth. 2019. Knowsemmlm: A knowledge infused semantic language model. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 550–562.
- Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. *arXiv preprint arXiv:1606.05679*.
- 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.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. [Event2Mind: Commonsense inference on events, intents, and reactions](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–473, Melbourne, Australia. Association for Computational Linguistics.

Robert P. Abelson Roger C. Schank. 1977. *Scripts, Plans, Goals, and Understanding : an Inquiry into Human Knowledge Structures*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

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.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. [Structured use of external knowledge for event-based open domain question answering](#). In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03*, page 33–40, New York, NY, USA. Association for Computing Machinery.

A Appendix: Prompt and Completion

A.1 Appendix A: Prompt and example completions for Step Generation

Prompt: List the events before an attack: 1.

Completion:

The attacker gathers information about the target.

2. The attacker plans the attack.
3. The attacker gains access to the target system.
4. The attacker executes the attack.
5. The attacker covers their tracks.

Prompt: List the events after an attack: 1.

Completion:

The attacker’s identity is confirmed.

2. The target is notified of the attack.
3. The attacker is placed on a watch list.
4. The attacker’s device is seized.
5. The attacker is arrested.

Prompt: List the sub-events involved in an attack:

1.

Completion:

Identify the target.

2. Plan the attack.
3. Choose the weapons.
4. Assemble the team.
5. Launch the attack.
6. Evaluate the results.

A.2 Appendix B: Prompt and example completions for Node Extraction

Prompt:

/ instruction */*

For each sentence, extract event verbs and their arguments, categorizing the arguments as subject or object. Write None if there is no object.

Return in [verb: _, subject: _, object: _] format.

/ few-shot examples */*

For example:

Q: Isaac ate a cake today and he played football.

A: [verb: eat, subject: Isaac, object: cake], [verb: play, subject: Isaac, object: football]

Q: The teacher arrived in class and he started teaching.

A: [verb: arrive, subject: teacher, object: class], [verb: start, subject: teacher, object: teaching]

Q: Nate and Isaac ate dinner.

A: [verb: eat, subject: Nate and Isaac, object: dinner]

Q: Justin slept.

A: [verb: sleep, subject: Justin, object: None]

/ target example */*

Q: The attacker gathers information about the target.

A:

Completion:

[verb: gather, subject: attacker, object: information]

/ target example */*

Q: The attacker’s identity is confirmed.

A:

Completion:

[verb: confirm, subject: attacker’s identity, object:

None]

/* target example */

Q: The attacker is placed on a watch list.

A:

Completion:

[verb: place, subject: attacker, object: watch list]

A.3 Appendix C: Prompt and example completions for Node Grounding

Prompt:

/* few-shot examples */

List event names related to the event "People are infected with this disease":

- 1.infection
- 2.epidemic
- 3.pandemic

List event names related to the event "It was a robbery-related incident":

- 1.robbery
- 2.burglary
- 3.theft

List event names related to the event "The first case of the disease have detected and it has been reported":

- 1.infection
- 2.epidemic
- 3.pandemic

List event names related to the event "The disease is eventually brought under control":

- 1.control
- 2.improvement

List event names related to the event "People who are ill have serious symptoms":

- 1.symptoms

List event names related to the event "The pathogen begins to spread through the population":

- 1.transmission
- 2.spread

/* target example */

List event names related to the event "The attacker gathers information about the target":

Completion:

- 1.reconnaissance
- 2.surveillance

3.investigation

/* target example */

List event names related to the event "The attacker's identity is confirmed":

Completion:

- 1.identification
- 2.confirmation

/* target example */

List event names related to the event "The attacker is placed on a watch list":

Completion:

- 1.surveillance
- 2.monitoring
- 3.investigation

PersLEARN : Research Training through the Lens of Perspective Cultivation

Yu-Zhe Shi^{1*}✉, Shiqian Li^{1,2*}, Xinyi Niu^{1,3*}, Qiao Xu^{1,3*}, Jiawen Liu^{1,4*}, Yifan Xu^{1,5*},
Shiyu Gu¹, Bingru He^{1,6}, Xinyang Li^{1,6}, Xinyu Zhao^{1,6}, Zijian Zhao^{1,6}, Yidong Lyu^{1,7}, Zhen Li^{1,8},
Sijia Liu^{1,9}, Lin Qiu^{1,10}, Jinhao Ji¹, Lecheng Ruan¹¹✉, Yuxi Ma¹¹, Wenjuan Han¹²✉, Yixin Zhu²✉

¹PersLab Research ²Peking University ³Huazhong University of Science and Technology ⁴Tongji University ⁵Kyushu University
⁶Tsinghua University ⁷Chinese University of Hong Kong ⁸University of Chinese Academy of Sciences ⁹University of the Arts London
¹⁰University of Hong Kong ¹¹National Key Laboratory of General Artificial Intelligence, BIGAI ¹²Beijing Jiaotong University
*Equal contributors ✉ syz@perslab.co, ruanlecheng@bigai.ai, wjhan@bjtu.edu.cn, yixin.zhu@pku.edu.cn

Abstract

Scientific research is inherently shaped by its authors’ perspectives, influenced by various factors such as their personality, community, or society. Junior researchers often face challenges in identifying the perspectives reflected in the existing literature and struggle to develop their own viewpoints. In response to this issue, we introduce PersLEARN, a tool designed to facilitate the cultivation of scientific perspectives, starting from a basic seed idea and progressing to a well-articulated framework. By interacting with a prompt-based model, researchers can develop their perspectives **explicitly**. Our human study reveals that scientific perspectives developed by students using PersLEARN exhibit a superior level of logical coherence and depth compared to those that did not. Furthermore, our pipeline outperforms baseline approaches across multiple domains of literature from various perspectives. These results suggest that PersLEARN could help foster a greater appreciation of diversity in scientific perspectives as an essential component of research training. ¹

1 Introduction

The pursuit of science is driven by a desire to gain a deeper understanding of the natural world, not only through the collection of objective facts but also through interpreting those facts (Kuhn, 1970; Longino, 1990). As a result, scientific knowledge is shaped by a complex interplay of various factors that extend beyond the objective world. These factors include the personal characteristics of individual scientists (Heisenberg, 1958; Bybee, 2006), shared mindsets within scientific communities (Cetina, 1999), and broader societal contexts such as cultural and political influences (Latour and Woolgar, 1986; Latour, 1987; Lynch, 1993; Latour et al., 1999). Together, these factors contribute to

¹Website: <https://perslearn.com/>. Video: <https://vimeo.com/802213150>.

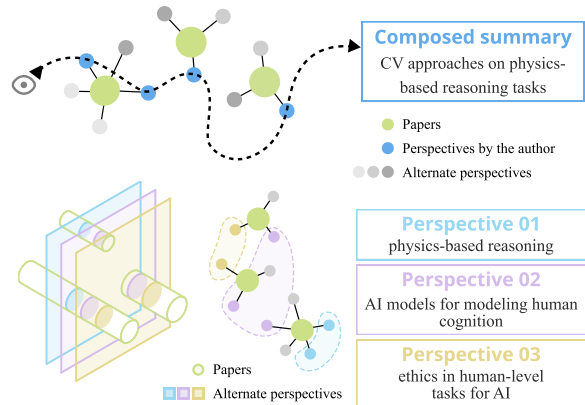


Figure 1: **Composed summaries vs. framed perspectives.** Composed summaries are subject to the authors’ perspectives, whereas the perspective frames are directed by new ideas.

forming perspectives regarding how best to interpret the natural world. Perspectives are essential to effectively process and communicate scientific knowledge with limited cognitive resources (Lewis et al., 2014; Griffiths et al., 2015; Gershman et al., 2015; Lieder and Griffiths, 2020).

However, junior researchers often face difficulties in developing their own scientific perspectives. They may struggle to identify the perspectives reflected in the existing literature and consequently struggle to develop and articulate their own viewpoints. This presents a significant obstacle to the progress of research training and deprives junior researchers of the opportunity to embrace the broader range of diverse perspectives that could contribute to their understanding of a particular topic (Duschl and Grandy, 2008). The challenge of developing scientific perspectives is particularly evident in one of the most significant research training approaches—writing literature reviews. In our pilot study, we asked students studying at the intersection of Artificial Intelligence (AI) and Cognitive Reasoning (CoRe) to write a review article from the perspective of “physics-based reasoning in Computer Vision (CV)” using a set of papers published on CV conferences. The assigned task aims to provide students with a multifaceted

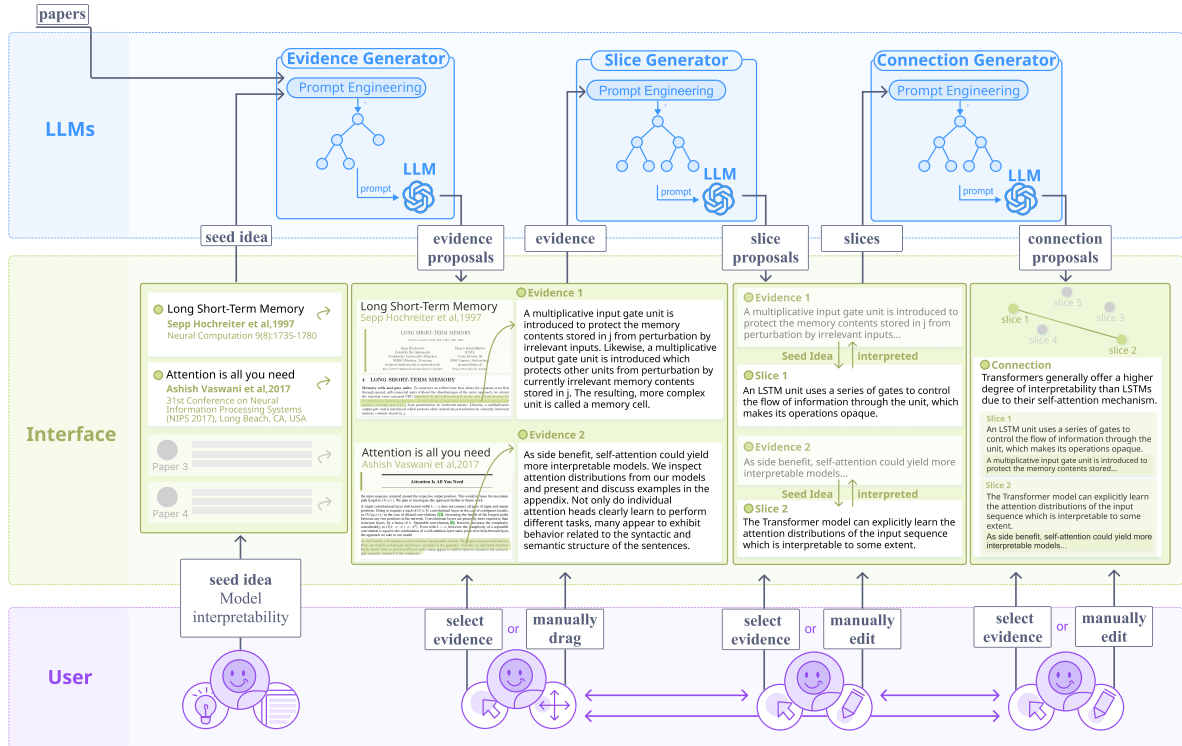


Figure 2: **The interactive workflow of PersLEARN**. This example showcases the scenario that a PersLEARN user intends to frame a rather novel perspective of “model interpretability” given the original papers on LSTM and Transformer. While the seed idea is not the major focus of both papers, evidence can be found to support that seed idea with the help of PersLEARN. The evidence includes the introduction of gates in LSTM to protect memory contents and the self-attention mechanism in Transformer, which can yield more interpretable models. Then the user re-interprets the evidence that LSTM operations are opaque due to the use of gates, while Transformer models have some level of interpretability through their attention distributions. It concludes that Transformers generally offer a higher degree of interpretability compared to LSTMs due to their self-attention mechanism. The process is assisted by prompt-engineered LLMs but is exactly determined by the user.

perspective on both computer vision and physics. Interestingly, most of the reviews the students composed do not have their own perspective; their reviews are titled “CV approaches on physics-based reasoning tasks” or have similar titles. This suggests that most students simply wrote summaries of every citing paper without considering an alternative perspective (see Fig. 1). To address this gap in research training, we propose PersLEARN, a tool that **explicitly** guides the process of cultivating scientific perspectives.

PersLEARN is grounded in classical theories drawn from the fields of cognitive and social sciences, particularly in the domain of scientific knowledge representation (Sec. 2.1). It provides an entire life-cycle of constructing a perspective frame that semi-automates researchers to start from a single seed idea and then iteratively interpret and structure relevant literature (Sec. 2.2). This process is facilitated through an interactive system that employs a hierarchical prompt-based approach to propose potential interpretations and structures based on a seed idea (Sec. 2.3). Experiments on both hu-

man evaluation (Sec. 3) and automatic evaluation of each module (Sec. 4) suggest that PersLEARN has the potential to enhance the quality of scientific research training significantly.

2 Design and Implementation

Designing PersLEARN is required to answer two questions: (i) What is the appropriate representation of perspective frames that makes researchers comfortable? (ii) How to informationize such representation for both user input and automated generation? In response to the questions, we highlight how PersLEARN is implemented from a theoretical framework to an interactive system step by step.

2.1 Theoretical Framework²

Following the principle of analogical education (Thagard, 1992; Aubusson et al., 2006), we create a system of analogies to ground the abstract concepts about perspectives. First, the scientific knowledge covered by the literature about a seed idea is in a

²View an abstract video illustration of the framework: <https://vimeo.com/802213146>.

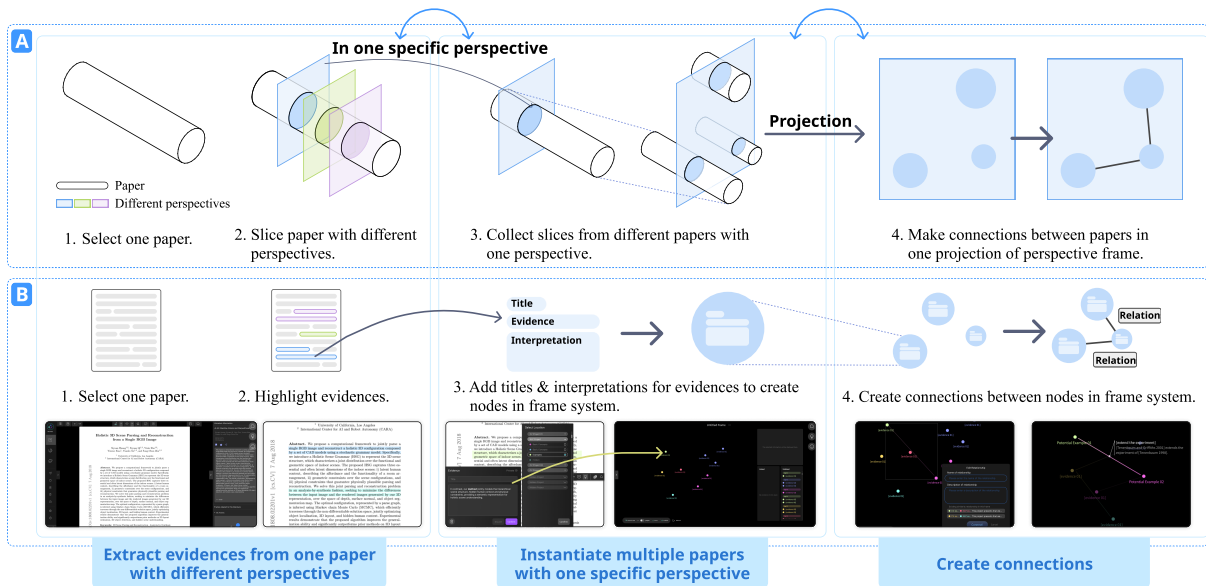


Figure 3: **Illustration of perspective cultivation.** (A) Visual analogy of the process: interpreting the evidence in the papers given the seed idea and structuring the papers with the relations between them. (B) User interfaces during the process.

higher-dimensional space than the perspective of a single paper (Duschl and Grandy, 2008). Here we set scientific knowledge of a seed idea as a 3D space and the specific perspective as a 2D plane for readability. For example, the seed idea “CV approaches on physics-based reasoning tasks” on the intersection of physics and CV can be framed as different specific perspectives, such as “physics-based reasoning” (Zhu et al., 2020), “AI models for modeling human cognition” (Lake et al., 2017), “evaluation metrics of new tasks in AI” (Duan et al., 2022), “ethics in human-level tasks for AI” (Jackson Jr, 2018), and “interpretability of physics-based reasoning AI models” (Edmonds et al., 2019). To not be trapped in a single perspective, we should pay attention to the ingredients of the papers rather than the ideas claimed by the authors. On this basis, framing another perspective is projecting the 3D space to another 2D plane by making *slices* from the papers, where each slice is a subset of ingredients. Such slices are articulated with others under the logic of the seed idea. Thus, a perspective frame is cultivated on the plane, growing from a seed idea with few slices to a graph with slices connected (see Fig. 3 for details).

Formally, the perspective frame is organized as a graph with information in nodes and edges on a 2D plane that instantiates the seed idea from the 3D space of scientific knowledge. The elements in a perspective frame can be described as follows:

- **Seed idea:** A rough textual description of the perspective, e.g., “Physics-based reasoning us-

ing CV approaches,” which serves as the starting point of the literature review and should be determined at the very beginning.

- **Evidence:** A piece of evidence comes from every paper in the selected set of literature, which contains the grounded information (a text span) supporting the given seed idea.
- **Slice:** A slice is the textual *interpretation* conditioned on the given seed idea based on a piece of evidence. A slice is a node in the graph.
- **Connection:** A connection between two slices is the textual interpretation conditioned on the perspective given the *relation* (e.g., relations-in-common such as *inspire* and *parallel*; and relations-of-distinction such as *improve*, *alternate*, and *compete*) between two slices. A connection is an edge in the graph.

Fig. 2 shows the interactive workflow of PersLEARN. Suppose one concerns the “*model interpretability*” (seed idea) of LSTM and Transformer, which is not the major perspective of either original paper of the two models. Given the corresponding two papers ‘*Long short-term memory*’ and ‘*Attention is all you need*’, the **evidence generator** finds the evidence to support the seed idea from the papers: ‘*A multiplicative input gate unit is introduced to protect the memory contents stored in j from perturbation by irrelevant inputs. Likewise, a multiplicative output gate unit is introduced which protects other units from perturbation by currently irrelevant memory contents stored in j . The resulting, more complex unit is called a memory cell.*

and ‘As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, but many also appear to exhibit behavior related to the syntactic and semantic structure of the sentences.’ The **slice generator** then generates the interpretations: ‘An LSTM unit uses a series of gates to control the flow of information through the unit, which makes its operations opaque.’ and ‘The Transformer model can explicitly learn the attention distributions of the input sequence which is interpretable to some extent.’ The **connection generator** finally provides the connection between these slices: ‘Transformers generally offer a higher degree of interpretability than LSTMs due to their self-attention mechanism.’ Such cultivation of a brand new perspective helps students *think outside the box*, which usually yields innovation in scientific research and should serve as one of the major parts in research training.

Notably, elements such as evidence, slices, and connections are not determined at once but may be revised in multiple iterations. As the perspective frame grows, the researcher’s understanding of the seed idea goes deeper, and the contents of slices and connections are sharpened accordingly. Hence, instead of answering a *chicken-or-the-egg* problem between slices and connections, our users generate them iteratively. Varied by the seed ideas, a perspective can be a well-organized collection of information (e.g., “performance comparison between backbone models on physical-reasoning tasks” (Duan et al., 2022)), a statement (e.g., “intuitive physics may explain people’s ability of physical reasoning” (Kubricht et al., 2017)), or a problem (e.g., “physical reasoning by CV approaches” (Zhu et al., 2020)). Though coming with different levels of abstraction, they all bring information gain, more or less (Abend, 2008).

PersLEARN well echoes the established theories, suggesting our design’s integrity. In a perspective frame, elements are contextualized in the entire frame by connecting with each other (Grenander, 2012; Shi et al., 2023); no element’s meaning is determined solely by itself. Moreover, any revision of an element influences the larger structure. Such representation has been shown as an innate knowledge representation of humans—*theory theory* (Gopnik, 1994; Gopnik and Meltzoff, 1997;

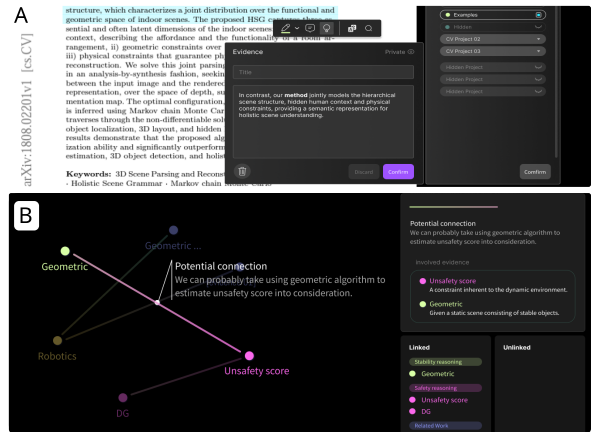


Figure 4: **UI showcases.** (A) A selected piece of evidence and its interpretation. (B) A generated perspective frame.

Carey, 1985, 2009). Furthermore, Carey (1986, 2000) have shown that such a framework can be captured and gradually revised by young students in terms of science education. To the best of our knowledge, current tools for literature review composing (e.g., ResearchRabbit, Connected Paper, Inciteful, and litmaps) all focus on visualizing literature relationships based on similarity and citation relationships without explicitly considering the framing of diverse perspectives.

2.2 Implementing User Interface (UI)

A researcher may develop a seed idea when reading a few papers, even if it is far from a mature perspective. The user first locates the evidence in a paper by dragging the mouse to select the text span through the PDFViewer and adds the selected span into Evidence Hub. Next, the user could generate a slice by writing a textual interpretation of the paper based on the evidence; this would trigger the initialization of a new perspective frame, and the first slice can be dragged into the canvas (implemented by D3.js library (Bostock et al., 2011)). The user can get back to the papers for more pieces of evidence and back to revising interpretations by clicking on the slices and editing the information at the right bar. With more than one slice in the canvas, the user can connect two slices by dragging the mouse around them and then write a textual interpretation of the relation between them. Likely to edit the slices, the user can also edit the connections by clicking on them and editing the information at the right bar. The perspective frame is cultivated by repeating these steps, buliding up the mindset for perspective framing in the *learning by doing* principle (Schank et al., 1999). Please refer to Fig. 4 for an exemplar perspective frame.

In the user-centered design of UI (Zaina et al.,

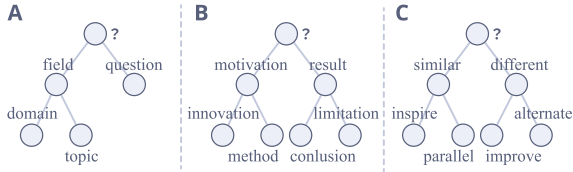


Figure 5: **Prompt Engineering.** (A) Evidence location. (B) Slice generation. (C) Connection generation.

2021), we follow the established theories in design for education (Hu et al., 1999; Miraz et al., 2016), such as color classification (Wen, 2021) and hierarchical information display (Jinxian, 2020). These support the integrity of our UI design.

2.3 Semi-automating the Procedure

We employ a hierarchical prompt-based approach to semi-automate the slice generation and connection generation. PersLEARN automatically generates some candidate proposals of slices and connections, and users can choose to accept, delete or modify these proposals.

Generate proposals of slices Scientific papers generally have similar and main-streaming structures (Dumont, 2014). Humans read scientific papers effectively while considering this prior structure rather than browsing aimlessly. We leverage this intuition by proposing the hierarchical prompt-based approach. This approach takes the seed idea and partial texts of the paper (*i.e.*, Abstract, Introduction, Discussion, and Conclusion) sections as input, and outputs the proposals of slices. We designed a hierarchical prompt-based approach (see Appx. A.1). First, we parse the seed idea to identify the specific field and domain of interest and fit the parsed terms into the prompting schema. Next, the prompted Large Language Model (LLM) extracts sentences from papers as evidence proposals. The LLM generates slice proposals conditioned on the evidence.

Specifically, it consists of two prompting stages: prompt generation and answer extraction. In the first stage, we first prompt an LLM with a generated prompt. After the LLM generates a response, we extract the information as the answer. Next, we traverse the hierarchical prompting schema from the top down to adopt a prompt template. Finally, we concatenate it with the texts of the paper as the prefix to generate the response. In the second stage, we post-process the response by removing repeated words and punctuation marks such as extra spaces.

Generate proposals of connections Similar to slice generation, generating proposals of connections follows a prompt-based approach. The

LLM takes two slices as input and outputs the relation between these two slices. A connection shows the relation between two slices (*e.g.*, relations-in-common such as inspire and parallel; and relations-of-distinction such as improve, alternate, and compete). Hence, we design the prompt as a multiple-choice question.

Our approach avoids uncontrollable and time-consuming manual designing while achieving comparable performance compared to existing fully-manual methods. Since we use the zero-shot setting, labor-consuming labeling is not required.

3 Human Evaluation

To validate PersLEARN for research training, we conducted a human study following the standard protocols of digital device auxiliary scenarios in higher education (Van den Akker, 1999; Neuman, 2014). This study is approved by the Institutional Review Board (IRB) of Peking University.

3.1 Method

Materials We created a scenario that simulates the training on writing literature reviews. The literature used in our simulation is five papers published at computer vision conferences. These papers have different topics varying from 3D scene parsing and reconstruction to learning object properties and using tools. However, they can be integrated together by interpreting from a physics-based perspective.

Participants We recruited 24 participants from the Peking University participant pool (11 female; mean age = 22.63). Every participant was paid a wage of \$14.6/h. We evenly divided participants into the control and experimental groups.

Procedures All participants were required to read the five papers and compose a short paragraph of literature review given the perspective “Physics-based reasoning.” Only the abstract, introduction, and conclusion/discussion were mandatory to read to reduce workload. The experiments lasted for 1 hour. The control group followed the standard procedure of writing reviews without PersLEARN as researchers usually do in their studies: reading the raw papers and writing the review. The experimental group utilized PersLEARN to create the review: locating evidence, interpreting, illustrating relations, and synthesizing the review. All participants were free to use the internet for extra help, such as searching for new concepts and unfamiliar words.

3.2 Result

We evaluate PersLEARN both quantitatively and qualitatively to verify whether it helps students compose more logical and pertinent reviews.

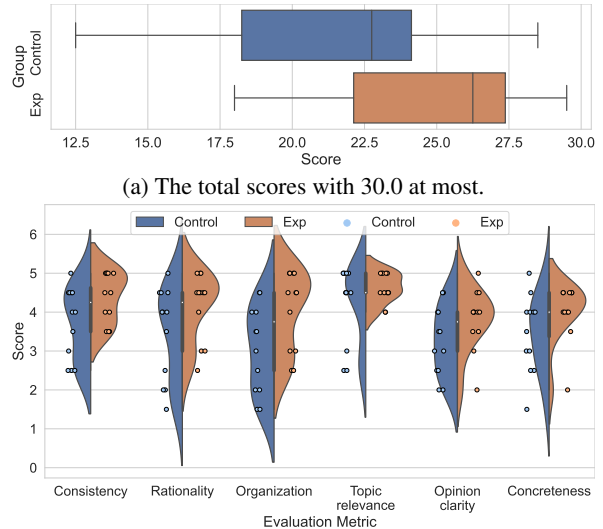
Quantitative evaluation The reviews from the control and experimental groups were shuffled and sent to experts to grade. The grading metrics include logicity and pertinence. Specifically, we asked 3 experts to grade on consistency (Farkas, 1985), rationality (Kallinikos and Cooper, 1996), organization (Kallinikos and Cooper, 1996), topic relevance (Hayes, 2012), opinion clarity (Williams, 1990), and concreteness (Sadoski et al., 2000); each ranks from 1 to 5. All of the experts hold Ph.D. degrees in related fields of AI, have been working on AI for at least eight years, and have no conflict of interest with the authors of this paper.

The average scores of the control and experimental groups are 21.25 and 25.08, respectively. Fisher’s exact test on the two variables (*i.e.*, whether PersLEARN was used and the score) reveals that the experimental group significantly outperforms the control group in both logicity and pertinence ($P = 0.0361$; see Fig. 6a), suggesting participants exploit the interpretations from a particular perspective and organize them by inducing their relations. Such a paradigm equips them with improved research training. Detailed scores on 6 evaluation metrics are shown in Fig. 6b. The results demonstrate a noticeable improvement in academic review writing in terms of logicity and pertinence for the experimental group; the experimental group’s performance shows a clear shift towards higher scores.

Qualitative evaluation We further conducted an interview to record qualitative comments after the participants in the control group finished their experiments. We interviewed them on how PersLEARN contributed to reading papers and composing reviews; see Appx. B.3 for interview questions. Most participants stated that PersLEARN helped them better understand the content of articles, think more clearly, and organize their writing expediently. For future work, they hoped to embed intelligent agents to provide protocols for each procedure.

3.3 Discussion

We present a case study to show how the experimental group composes better reviews than the control group; see representative paragraphs in



(a) The total scores with 30.0 at most.
(b) The scores of the 6 metrics with 5.0 for each. The presented scores in the plot may surpass 5.0 due to smoothing.

Figure 6: **The scores of the control and experimental groups.** The scores of each review are averaged across experts.

Appx. C.1. We conclude from our human study that PersLEARN can boost literature reading and review writing by providing a perspective-guided thinking framework of evidence locating, interpretation deriving, and relation inducing.

4 Automatic Evaluation

To automatically evaluate PersLEARN at scale, we introduce a *perspective reconstruction* task with three sub-tasks (slice generation, connection generation, and diversity evaluation), requiring the system to recover an established perspective frame given the same seed idea.

4.1 Benchmark Construction

Dataset Collection We carefully construct a testing set with reputation-established narrative reviews, expert reviews, and opinion articles to obtain a high-quality ground truth of perspectives. Systematic reviews and articles of information collection are removed from the set because such papers do not provide a sharp and unique perspective; we ensure that all articles are developed around a concrete and coherent perspective. Moreover, we ensure that every title is the epitome of the perspective held by the article; we treat the titles as seed ideas.

36 review articles are collected from diverse domains standing at the intersection of AI and CoRe, including CV, Natural Language Processing (NLP), Intuitive Physics (Phy), Causality (Cau), Abstract Reasoning (AbsRe), Mirroring and Imitation (MrIm), Tool Use (Tool), Non-verbal Communication (NvComm), Intentionality (Int), Theory

Table 1: **Result of our pipeline.** w/o and w/ are with and without prompt engineering, respectively. The performance of slice generation, connection generation, and perspective diversity indicate the efficacy of our prompt engineering.

Metric	w/o	w/
Slice BLEURT	0.238	0.795
Connection CR	0.450	0.550
Perspective VMR	0.028	0.006

of Mind (ToM), and Utility (U). This generates a literature set with 333 papers cited by at least one of the articles. Among these, 24 papers are cited by more than one article. Some of the papers are directly obtained from S2ORC (Lo et al., 2020), while others are parsed from raw PDF.

Evaluation Metrics of slice generation For a cited paper in the original review, we treat the coherent sentences around the citation mark as the ground truth for the corresponding slice, following the same protocol as in Li et al. (2022). Because the semantic meaning is critical (rather than the wording and phrasing), we employ BLEURT (Selam et al., 2020) rather than word-wise evaluation metrics like ROUGE and BLEU (Lin, 2004; Papineni et al., 2002). BLEURT score indicates the similarity between two statements; larger scores mean better performance.

Evaluation Metrics of connection generation Since the connection between two papers under the same perspective is only conditioned on the slices, we focus on the logical consistency between the generated connection and the two input slices. Following the setting of Natural Language Inference (NLI), we calculate the Consistent Rate (CR), the proportion of entailment prediction in all predictions. Higher CR indicates better performance. We employ the state-of-the-art model, DeBertaV3 (He et al., 2021b,a), as the NLI model for evaluation.

Evaluation Metrics of Diversity This is an extended case study based on slice generation. We specially study how different perspectives drive the interpretations from the same set of papers. We calculate the normalized Variance-to-Mean Ratio (VMR) over the BLEURT scores on all established perspectives of a set of papers for each approach. Lower VMR indicates that an approach generates slices conditioned on different perspectives well.

4.2 Experiments of Slice Generation

Setup We use **InstructGPT** as the backbone LLM model (Ouyang et al., 2022) for our prompt-based approach. The input and output are the same as in Sec. 2.3. The baseline approach directly

prompts the LLM with the target output without the proposed hierarchical prompting schema.

Results The BLEURT results in Tab. 1 show that the generation with prompt engineering outperforms that without by a large margin (233%). This result validates our pipeline in abstract understanding and perspective-based interpreting; see representative slices in Appx. C.2.

4.3 Experiments of Connection Generation

Setup We use **InstructGPT** as the backbone LLM model. The input and output of this evaluation are the same as the connection proposed in Sec. 2.3. The baseline approach directly prompts the LLM with the target output.

Results As shown in Tab. 1, our connection generation module surpasses the baseline approach in CR by a large margin (22%). It means more logical connections are generated by our approach and thus contribute to more entailment predictions. See representative connections in Appx. C.3.

4.4 Experiments on Diverse Perspectives

Setup We use **InstructGPT** as the backbone LLM model for both the slice and the connection generation modules. The baseline approach adopts the slice and connection generation modules without the proposed schema.

Results The VMR results in Tab. 1 show that PersLEARN generates slices of richly diverse perspectives, surpassing the baseline by a large margin (79%). We present some examples of the interpretations of different perspectives; see representative slices in Appx. C.4.

5 Discussion

We present PersLEARN to facilitate scientific research training by explicitly cultivating perspectives. Human study shows that PersLEARN significantly helps junior researchers set up the mindset for jumping out of perspective given by the literature and framing their own ones. Extensive benchmarking shows that our system has the potential to mine perspectives out of diverse domains of literature without much human effort. These experiments suggest that PersLEARN has the potential to support scientific research training in general—from explicating one’s own perspective to embracing the diverse perspectives of others. Readers can refer to the “Broader Impact” and “Limitation” sections (Sec. 5) for further discussions.

Ethics Statement

The human study presented in this work has been approved by the IRB of Peking University. We have been committed to upholding the highest ethical standards in conducting this study and ensuring the protection of the rights and welfare of all participants. Considering that the workload of the procedure for participants is relatively high among all human studies, we paid the participants a wage of \$14.6/h, which is significantly higher than the standard wage (about \$8.5/h). Every expert was paid \$240 for grading the 24 review paragraphs composed by the participants.

We have obtained informed consent from all participants, including clear and comprehensive information about the purpose of the study, the procedures involved, the risks and benefits, and the right to withdraw at any time without penalty. Participants were also assured of the confidentiality of their information. Any personal data collected (including name, age, and gender) was handled in accordance with applicable laws and regulations.

Broader Impact

The underlying impact of the mindset brought by PersLEARN goes beyond research training toward science education in general. Specifically, PersLEARN provides the infrastructure for further investigation in two aspects: (1) embracing the diverse perspectives of the same scientific topic to construct a stereoscopic understanding of the topic; (2) facilitating the communication between junior researchers with different mindsets.

The broader impact is analogous to the classic fable *Blind men and an elephant*, where each man interpreted the elephant differently because they were standing on different perspectives. Though this has been a metaphor complaining that science is limited by observation (Heisenberg, 1958), it highlights the virtue of scientific research—focused, and every young researcher understands and interprets science from a focused perspective. Hence, to gain a more comprehensive view of the elephant, the blind men may put their understandings of it together and then try to synthesize it based on their perspectives. In contrast, a sighted person may view the elephant from a distance and capture a holistic view at first—she ends up with a superficial understanding of the elephant if not selecting a perspective and going close to the elephant, like the blind. Thus, by embracing diverse perspectives (*i.e.*, visualizing the

perspective frames in a hub), one gets a stereoscopic view and, more importantly, a deeper understanding of the scientific topic. Moreover, when the metaphorical *blind men* in the fable attempt to articulate their distinct perspectives, they may be hindered by the gap between mindsets. To exemplify, individual might struggle to comprehend the concept of a “fan”, which in their perception, the elephant appears to resemble. This suggests that the communication of science should be executed in a listener-aware way and that the speaker’s perspective should be transformed (*i.e.*, by changing the terms used in slices and connections) to its analogical equivalent in the listener’s mindset. Thus, science can be communicated easily, facilitating its transparency, reliability, and the chances of cross-domain collaboration. In summary, our framework of scientific perspective may bring science education to a future with better student-centered considerations (Leshner, 2018).

Limitations

As a preliminary work, the design and evaluation of PersLEARN come with limitations, leading to further investigations:

- Can we construct a larger scale dataset of explicit perspective frames of the literature for more fields in the sciences, such as biology, sociology, *etc.*?
- Can we fine-tune LLMs on the larger dataset to obtain better performance on slice and connection generation?
- Can we carry out a human study at a larger temporal scale, say during one semester, to track the progress of students using PersLEARN ?

With many questions unanswered, we hope to facilitate research training and science education in a broader way.

Acknowledgements

The authors would like to thank Dr. Yujia Peng, Yuxi Wang, and Zili Li for their kind help in setting up the human study. This work is supported by the Institute for Artificial Intelligence at Peking University. W.H. is supported by the Talent Fund of Beijing Jiaotong University (2023XKRC006). All authors declare no conflict of interest.

References

- Gabriel Abend. 2008. The meaning of ‘theory’. *Sociological Theory*, 26(2):173–199.
- Peter J Aubusson, Allan G Harrison, and Stephen M Ritchie. 2006. *Metaphor and analogy: Serious thought in science education*. Springer.
- Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graph*, 17(12):2301–2309.
- Rodger W Bybee. 2006. Scientific inquiry and science teaching. *Scientific inquiry and nature of science: Implications for teaching, learning, and teacher education*, pages 1–14.
- Susan Carey. 1985. *Conceptual change in childhood*. MIT Press.
- Susan Carey. 1986. Cognitive science and science education. *American Psychologist*, 41(10):1123.
- Susan Carey. 2000. Science education as conceptual change. *Journal of Applied Developmental Psychology*, 21(1):13–19.
- Susan Carey. 2009. *The Origin of Concepts*. Oxford University Press.
- Karin Knorr Cetina. 1999. *Epistemic cultures: How the sciences make knowledge*. Harvard University Press.
- Jean-luc Doumont. 2014. Structuring Your Scientific Paper. In *English Communication for Scientists*. Nature.
- Jiafei Duan, Arijit Dasgupta, Jason Fischer, and Cheston Tan. 2022. A survey on machine learning approaches for modelling intuitive physics. In *International Joint Conference on Artificial Intelligence*.
- Richard A Duschl and Richard E Grandy. 2008. *Teaching scientific inquiry: Recommendations for research and implementation*. BRILL.
- Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. 2019. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37).
- David K Farkas. 1985. The concept of consistency in writing and editing. *Journal of Technical Writing and Communication*, 15(4):353–364.
- Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. 2015. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278.
- Alison Gopnik. 1994. The theory theory. In *Mapping the mind: Domain specificity in cognition and culture*, pages 257–293. Cambridge University Press.
- Alison Gopnik and Andrew N Meltzoff. 1997. *Words, thoughts, and theories*. MIT Press.
- Ulf Grenander. 2012. *A calculus of ideas: a mathematical study of human thought*. World Scientific.
- Thomas L Griffiths, Falk Lieder, and Noah D Goodman. 2015. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 7(2):217–229.
- John Hayes. 2012. Modeling and remodeling writing. *Written communication*, 29:369–388.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.
- Werner Heisenberg. 1958. Physics and philosophy: The revolution in modern science. *Physics Today*, 11(9):36.
- Paul Jen-Hwa Hu, Pai-Chun Ma, and Patrick YK Chau. 1999. Evaluation of user interface designs for information retrieval systems: a computer-based experiment. *Decision Support Systems*, 27(1-2):125–143.
- Philip C Jackson Jr. 2018. Toward beneficial human-level ai... and beyond. In *AAAI Spring Symposia*.
- Lin Jinxian. 2020. Application of guiding design in ui design of mobile terminal. In *2020 IEEE International Conference on Innovation Design and Digital Technology (ICIDDT)*.
- Jannis Kallinikos and Robert Cooper. 1996. Writing, rationality and organization: An introduction. *Scandinavian Journal of Management*, 12:1–6.
- James R Kubricht, Keith J Holyoak, and Hongjing Lu. 2017. Intuitive physics: Current research and controversies. *Trends in Cognitive Sciences*, 21(10):749–759.
- Thomas S Kuhn. 1970. *The structure of scientific revolutions*. University of Chicago Press: Chicago.
- Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253.
- Bruno Latour. 1987. *Science in action: How to follow scientists and engineers through society*. Harvard University Press.
- Bruno Latour and Steve Woolgar. 1986. *Laboratory life: The construction of scientific facts*. Princeton University Press.

- Bruno Latour et al. 1999. *Pandora's hope: Essays on the reality of science studies*. Harvard University Press.
- Alan I. Leshner. 2018. Student-centered, modernized graduate stem education. *Science*, 360(6392):969–970.
- Richard L Lewis, Andrew Howes, and Satinder Singh. 2014. Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in Cognitive Science*, 6(2):279–311.
- Xiangci Li, Biswadip Mandal, and Jessica Ouyang. 2022. Corwa: A citation-oriented related work annotation dataset. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Falk Lieder and Thomas L Griffiths. 2020. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.
- Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S Weld. 2020. S2orc: The semantic scholar open research corpus. In *Annual Meeting of the Association for Computational Linguistics*.
- Helen E Longino. 1990. *Science as social knowledge: Values and objectivity in scientific inquiry*. Princeton University Press.
- Michael Lynch. 1993. *Scientific practice and ordinary action: Ethnomethodology and social studies of science*. Cambridge University Press.
- Mahdi H Miraz, Peter S Excell, and Maaruf Ali. 2016. User interface (ui) design issues for multilingual users: a case study. *Universal Access in the Information Society*, 15:431–444.
- Delia Neuman. 2014. Qualitative research in educational communications and technology: A brief introduction to principles and procedures. *Journal of Computing in Higher Education*, 26:69–86.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.
- 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*.
- Mark Sadoski, Ernest T Goetz, and Maximo Rodriguez. 2000. Engaging texts: Effects of concreteness on comprehensibility, interest, and recall in four text types. *Journal of Educational Psychology*, 92:85.
- Roger C Schank, Tamara R Berman, and Kimberli A Macpherson. 1999. Learning by doing. *Instructional-design theories and models: A new paradigm of instructional theory*, 2(2):161–181.
- Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleu: Learning robust metrics for text generation. In *Annual Meeting of the Association for Computational Linguistics*.
- Yu-Zhe Shi, Manjie Xu, John E. Hopcroft, Kun He, Joshua B Tenenbaum, Song-Chun Zhu, Ying Nian Wu, Wenjuan Han, and Yixin Zhu. 2023. On the complexity of bayesian generalization. In *International Conference on Machine Learning*.
- Paul Thagard. 1992. Analogy, explanation, and education. *Journal of Research in Science Teaching*, 29(6):537–544.
- Jan Van den Akker. 1999. Principles and methods of development research. *Design Approaches and Tools in Education and Training*, pages 1–14.
- Guoying Wen. 2021. Research on color design principles of ui interface of mobile applications based on vision. In *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*.
- Joseph Williams. 1990. *Toward clarity and grace. Chicago: The University of Chicago*.
- Luciana AM Zaina, Helen Sharp, and Leonor Barroca. 2021. Ux information in the daily work of an agile team: A distributed cognition analysis. *International Journal of Human-Computer Studies*, 147.
- Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, et al. 2020. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 6(3):310–345.

A Implementation Details

A.1 Prompting Schema

Slice generation We specifically engineer a prompting schema in a hierarchical fashion. First, we parse the seed idea to identify the specific field and domain:

- {seed idea} + What fields and domains does the article focus on? Only list the name.

Then, we use the following prompt to detect evidence:

- {paper} + Which sentences in the text are about {fields}? List the original sentences.

Finally, we match the following prompts with the parsed terms to generate an interpretation:

- {evidence} + What are the motivations in the text for studying {fields}?
- What methods and approaches in the text are used to study {fields}?
- what theories, models, and methods in the text are proposed to study {fields}?
- what results and conclusions in the text related to {fields} are drawn?
- what results and conclusions in the text related to {fields} are drawn?
- what implications or suggestions in the text for future research of {fields} are advocated?

Connection generation Our engineered prompt comes in a selective fashion.

- What are the differences (improve, alternate, compete) between the work {slice_1} and another work {slice_2} on motivations, methods, results, or conclusions to study {fields}?
- What are the similarities (inspire, parallel) between the work {slice_1} and another work {slice_2} on motivations, methods, results, or conclusions to study {fields}?

A.2 Alternate Approach

Slice generation The baseline prompt is simply a direct prompt.

- {evidence} + What interpretation about {fields} can we get from the text?

Connection generation The baseline prompt is simply a direct prompt.

- What are the differences and similarities between the work {slice_1} and another work {slice_2} on studying {fields}?

B Experimental Details

B.1 Instructions for Participants

Read the abstract, introduction, discussion, and conclusion sections of the following article, and write a short review entitled “Physics-based reasoning” in a txt file using the given process (see the tutorial for instructions; only for the experimental group). The experiments will last for one hour. You can use the Internet to help you write.

- 1 Holistic 3d Scene Parsing and Reconstruction from a Single RGB Image. ECCV, 2018.
- 2 Scene Understanding by Reasoning Stability and Safety. IJCV, 2015.
- 3 Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. NeurIPS, 2015.
- 4 Physics 101: Learning Physical Object Properties from Unlabeled Videos. BMVC, 2016.
- 5 Understanding Tools: Task-oriented Object Modeling, Learning, and Recognition. CVPR, 2015.

B.2 Interfaces in the Procedures

We show the screenshots of user interactions during the experiments, from entering an input perspective Fig. A1, to selecting papers Fig. A2, generating pieces of evidence Fig. A3, generating slices Fig. A4, generating connections Fig. A5, and browsing the perspective frame Fig. A6. To note, though these steps are demonstrated in a monotonic order here, every step is repeatable and extensible.

B.3 Interview questions

We interview participants on the following two questions.

- How does PersLEARN help you compose reviews?
- How can PersLEARN be improved?

C Extended Results

C.1 Perspective Paragraphs by Subjects

We show anonymized representative examples from both the control group and the experimental group of the human study. All examples are kept original without any revision, including typos. Colored texts are used to **highlight interpretation** and **relation** respectively facilitated by slices and connections of PersLEARN in the experimental group.

The top three paragraphs from the experimental group **with** pertinent interpretations and logical relations:

#1:

Physics-based reasoning has been used for two aims. **The first is to learn the physical properties of an object.** For example, Galileo[3] and Physics 101[4] learn **physical properties** like mass and density from videos. Bo Zheng et al[2] learn **stability and safety** of objects in a scene. **The second is to enrich the object representation by incorporating physical features.** The enriched representation is then used to assist other visual tasks. Siyuan Huang et al[1] design a physically enriched HSG representation of 3D scene structure in the single-view 3D reconstruction task. Yixin Zhu et al[5] use a representation consisting four **physical-functional components** in object recognition task.

#2:

Relatively brief runs of MCMC can drive [simulations in physics engine](#) to fit the key features of visual image, which has a similarly accurate outcome comparing with human intuitions[3]. [Further study expand](#) the abilities of learning the basic features of scenes, which makes the 3D parsing and reconstruction real. HSG can establish a joint distribution over the functional and geometric space of scenes, which captures the latent human context, geometric constraints and [physical constraints](#)[1]. [By implementing a new framework](#), visual system learns the [tools properties](#), the using methods, and the later action to do some related works, which not only recognize the appearance, but also explain the [physical mechanisms](#)[5].

#3:

Conventional scene understanding methods mostly neglect the object's [physical properties](#), rendering their weak ability of accurately understanding the scenes. [To address this issue](#), Zheng et al. [2] proposed a novel 3D scene understanding approach from a new perspective of reasoning object [stability and safety using intuitive mechanics](#). [As a step further](#), Huang et al. [1] proposed a computational framework to jointly parse a single RGB image and reconstruct a holistic 3D configuration, jointly considering latent human context, geometric constraints, and, physical constraints to guarantee the [physical plausibility](#).

The top three paragraphs from the control group which **fail to** interpret and organize from the perspective of physics-based reasoning:

#1:

The five papers all concerns over a main topic, that is how to effectively train artificial intelligence to perceive the outside physical world and afterwards giving different forms of feedback or guidance on new circumstances. The first and second are generally about understanding scenes but have some differences in their domains .The first one using a RGB Image to generate 3D scene applying the Markov chain Monte Carlo (MCMC) method[1], and the second focusing on building novel algorithms which are able to reason object stability and safety using intuitive mechanics with the representation of the disconnect graphs and disturbance field[2]. They all propose a new perspective for machines to logically and correctly process the human-understood information.

#2:

Machine learning and deep learning are applied to study physical object properties. In 2015, Jiajun Wu proposed a generative model for solving problems of physical scene understanding from real-world videos and images[3]. As the same time, Yixin Zhu presented a new framework – task-oriented modeling, learning and recognition which aims at understanding the underlying functions, physics and causality in using objects as "tools"[5]. Later in 2016, Jiajun Wu proposed an unsupervised model to learn physical object properties from unlabeled video[4]. Also, physics-based reasoning plays an important role in 3D parse and holistic 3D scene understanding. Bo Zheng presented a new perspective for 3D scene understanding by reasoning object stability and safety using intuitive mechanics[2]. Siyuan Huang proposed a computational framework to jointly parse a single RGB image and reconstruct a holistic 3D configuration composed by a set of CAD models using a stochastic grammar model[1].

#3:

The complexity and richness of human vision are not only reflected by the ability to recognize visible objects, but also to reason about the latent actionable information, including inferring physical object properties[4] or latent human context as the functionality of a scene[1]. But how does human vision achieve such complexity and richness ? Some studies propose a model based on physics principle for understanding these mechanisms[2,3,4]. Even other studies consider more complex situations, where humans have tasks[5] and other hidden human context[1].

C.2 Slices Generated by Machines

We present several examples of the generated slices with their BLEURT scores from the pipeline with and without prompt engineering.

#1: w/o: -1.132

The article suggests that deep learning, adversarial attacks, and countermeasures are independent of the data miner's activities.

#1: w/: 0.326

The article focuses on the motivations for studying deep learning, adversarial attacks, and countermeasures in order to better understand and defend against data manipulation by adversaries. The paper uses deep learning, adversarial attacks, and countermeasures to study the problem. The paper proposes a formal framework and algorithms for studying deep learning, adversarial attacks, and countermeasures. It also discusses theories, models, and methods related to these fields and domains. The article suggests that future research should focus on developing deep learning systems that are more robust to adversarial attacks and require less human intervention to keep up with them.

#2: w/o: -1.090

The article suggests that understanding the cognitive development, infant development, and developmental psychology of others is essential for successful social interactions.

#2: w/: 0.616

The article uses a variety of methods and approaches to study the fields of cognitive development, infant development, and developmental psychology. These include observational studies, experiments, and surveys. The article also uses a variety of theoretical frameworks, such as Piaget's theory of cognitive development, to analyze the data. The strengths of the text are that it provides evidence of infants' anticipatory responses to others' actions from an early age, and it also provides evidence that infants attend prospectively to others' actions. The limitations of the text are that it does not clarify whether infants anticipated the goal per se because the goal and pattern of movement were confounded.

#3: w/o: -0.735

Sentiment analysis, news categorization, question answering, and natural language inference are all techniques used to analyze text and extract meaning from it.

#3: w/: 0.422

The methods and approaches used to study sentiment analysis, news categorization, question answering, and natural language inference in the text include supervised learning, unsupervised learning, deep learning, natural language processing, and machine learning. The theories, models, and methods proposed to study sentiment analysis, news categorization, question answering, and natural language inference in the text are Convolutional Neural Networks (CNNs) and SVMs with well-engineered features.

C.3 Connections Generated by Machines

Our pipeline can discover the similarities and differences between slices. Below are some representative connections the model generates generated by the model with and without prompt engineering.

#1: w/o

The two works have similar motivations for studying Knowledge Graph Representation Learning, Knowledge Acquisition and Completion, Temporal Knowledge Graph, and Knowledge-Aware Applications. The methods used in the two works are also similar, with Knowledge Graph Representation Learning typically involving graph embedding, graph convolutional networks, and graph neural networks, Knowledge Acquisition and Completion typically involving rule-based approaches, ontology-based approaches, and machine learning-based approaches, Temporal Knowledge Graphs typically involving temporal graph embedding, temporal graph convolutional networks, and temporal graph neural networks, and Knowledge-Aware Applications typically involving knowledge-aware recommendation systems, knowledge-aware question answering systems, and knowledge-aware search systems. The results and conclusions of the two works are also similar, with Knowledge Graph Representation Learning theories, models, and methods including graph embedding, graph convolutional networks, graph attention networks, and graph generative models, Knowledge Acquisition and Completion methods including rule-based methods, ontology-based methods, and machine learning-based methods, Temporal Knowledge Graph methods including temporal graph embedding, temporal graph convolutional networks, and temporal graph attention networks, and Knowledge-Aware Applications including question answering, recommendation systems, and natural language processing.

#1: w/

The two works have similar motivations, methods, results, and conclusions when it comes to studying Knowledge Graph Representation Learning, Knowledge Acquisition and Completion, Temporal Knowledge Graph, and Knowledge-Aware Applications. The main difference between the two works is that the first work focuses on how knowledge graphs can be used to extract relations from text and can be used for knowledge acquisition and completion, while the second work provides an overview of the topics and suggests that they are related and can be used to develop applications that are aware of knowledge.

#2: w/o

The motivations for studying Artificial Intelligence, Cognitive Science, Pattern Recognition, Physics, and Psychology in the text are to gain a better understanding of how humans learn and process information, and to develop algorithms that can learn from fewer examples than traditional machine learning algorithms. The methods used in the text are theories, models, and methods from Artificial Intelligence, Cognitive Science, Pattern Recognition, Physics, and Psychology to study simple visual concepts. The results of the text suggest that future research should focus on understanding how people learn new concepts from just one or a few examples, and how people learn such abstract, rich, and flexible representations. The conclusion of the text is that any theory of concept learning should address these questions and provide implications or suggestions for future research in Artificial Intelligence, Cognitive Science, Pattern Recognition, Physics, and Psychology.

#2: w/

The two works have similar motivations in studying Artificial Intelligence, Cognitive Science, Pattern Recognition, Physics, and Psychology. The methods used in both works are similar, as they both suggest combining multiple levels of representation with the flexible use of long range context. The results of the first work suggest that deep Long Short-term Memory RNNs are a powerful model for sequential data and can be used for tasks such as cursive handwriting recognition and phoneme recognition. The results of the second work suggest that Artificial Intelligence, Cognitive Science, Pattern Recognition, Physics, and Psychology have not been able to replicate two aspects of human conceptual knowledge. The conclusions of both works are that combining multiple levels of representation with the flexible use of long range context can lead to improved results.

C.4 Slices for Different Perspectives

We show some interpreted slices of two cited papers (indexed by #) from different perspectives (*italicized*). These two papers are:

- 1 Heuristic judgment of mass ratio in two-body collisions. *Perception & Psychophysics* 56 (1994): 708-720.
- 2 Learning a theory of causality. *Psychological Review* 118.1 (2011): 110.

#1: *Intuitive Physics: Current Research and Controversies*

Intuitive Physics, Perception, Reasoning, and Artificial Intelligence are all studied in order to gain a better understanding of how the world works and how to create more efficient and effective systems. These fields are also studied in order to develop new technologies and applications that can be used to improve the lives of people. Methods and approaches used to study Intuitive Physics, Perception, Reasoning, and Artificial Intelligence include computational modeling, cognitive psychology, neuroscience, and machine learning. Theories, models, and methods proposed to study Intuitive Physics, Perception, Reasoning, and Artificial Intelligence include Bayesian inference, probabilistic graphical models, deep learning, reinforcement learning, and evolutionary algorithms. The results and conclusions drawn from the text related to Intuitive Physics, Perception, Reasoning, and Artificial Intelligence are that humans have an innate ability to understand physical concepts and use them to make decisions and solve problems. This suggests that humans have an intuitive understanding of physics that can be used to inform Artificial Intelligence algorithms. Additionally, the text suggests that humans are capable of making decisions and solving problems based on their perception of the physical world, and that this ability can be used to inform Artificial Intelligence algorithms.

#1: *Mind Games: Game Engines as an Architecture for Intuitive Physics*

The motivations for studying Artificial Intelligence, game development, and physics simulation are to gain a better understanding of how these technologies work, to develop new applications and technologies, and to explore the potential of these technologies for solving real-world problems. Artificial intelligence, game development, and physics simulation can be studied using a variety of methods and approaches, including machine learning, deep learning, reinforcement learning, evolutionary algorithms, and probabilistic methods. Artificial intelligence, game development, and physics simulation can be studied using a variety of theories, models, and methods. These include machine learning, deep learning, reinforcement learning, evolutionary algorithms, game theory, and physics-based simulations. Strengths of the text for studying Artificial Intelligence, game development, and physics simulation include its comprehensive coverage of the topics, its use of examples to illustrate key concepts, and its clear explanations of complex topics. Limitations of the text include its lack of in-depth coverage of certain topics and its lack of discussion of the latest developments in the field.

#2: *Bayesian Models of Conceptual Development: Learning as Building Models of the World*

The motivations in the text for studying Cognitive Development, Core Knowledge, Child as Scientist, Bayesian Program Induction, Computational Advances, Scientific Theories, Intuitive Theories, Biological Evolution, and Cultural Evolution are to gain a better understanding of the principles of causal reasoning and to develop a more comprehensive account of causality. Cognitive Development, Core Knowledge, Child as Scientist, Bayesian Program Induction, Computational Advances, Scientific Theories, Intuitive Theories, Biological Evolution, and Cultural Evolution are all methods and approaches used to study the blessing of abstraction. Cognitive Development, Core Knowledge, Child as Scientist, Bayesian Program Induction, Computational Advances, Scientific Theories, Intuitive Theories, Biological Evolution, and Cultural Evolution are all theories, models, and methods proposed to study Cognitive Development,

Core Knowledge, Child as Scientist, Bayesian Program Induction, Computational Advances, Scientific Theories, Intuitive Theories, Biological Evolution, and Cultural Evolution.

#2: *Intuitive Theories*

The motivations in the text for studying Cognitive science, psychology, and philosophy are to gain a better understanding of the principles of causal reasoning and to develop a description of the principles by which causal reasoning proceeds. Cognitive science, psychology, and philosophy are studied using methods and approaches such as logical reasoning, empirical observation, and experimentation. Cognitive science, psychology, and philosophy are studied using theories, models, and methods such as Bayesian networks, causal inference, and counterfactual reasoning. The results and conclusions drawn from the text related to Cognitive science, psychology, and philosophy are that abstract reasoning can be used to quickly learn causal theories, and that this can be beneficial in certain situations.

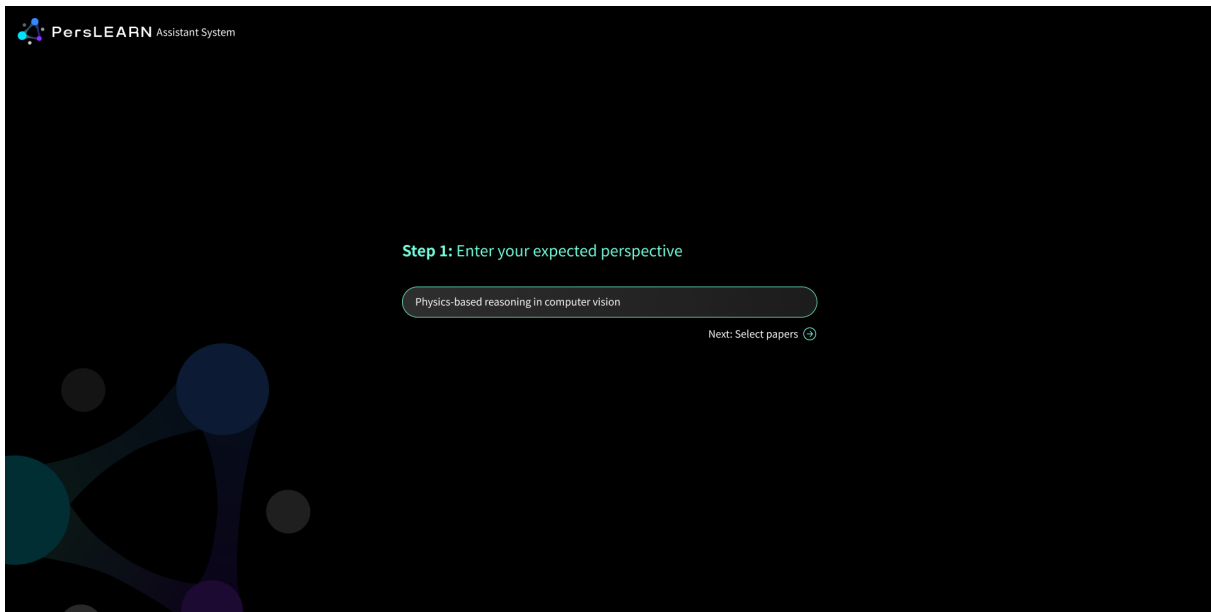


Figure A1: Input seed idea

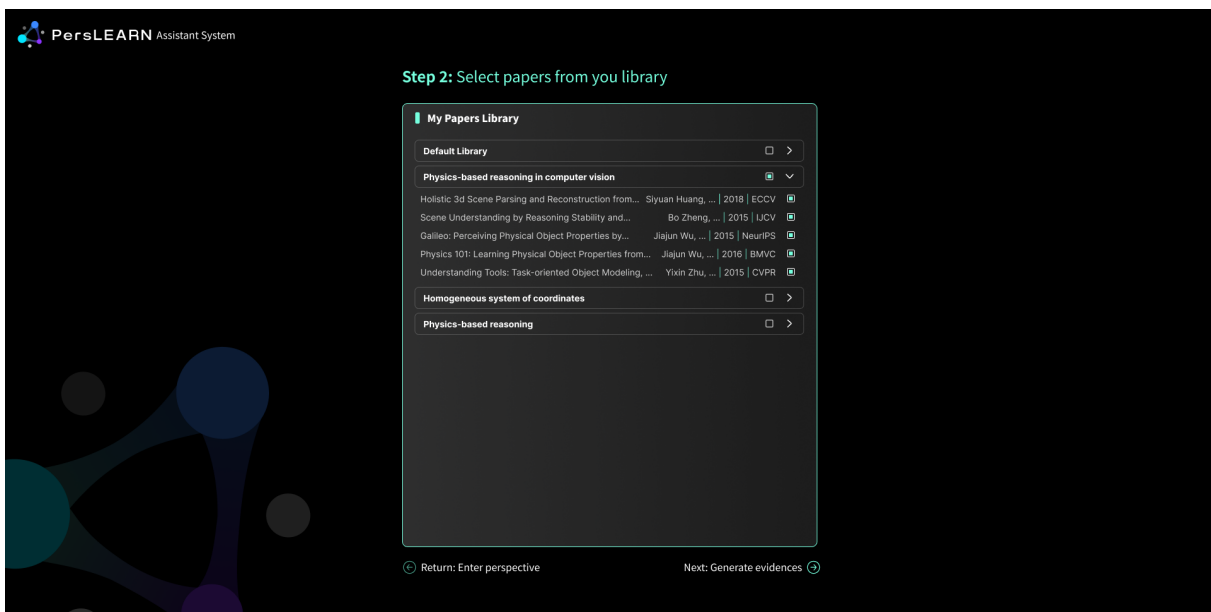


Figure A2: Select papers

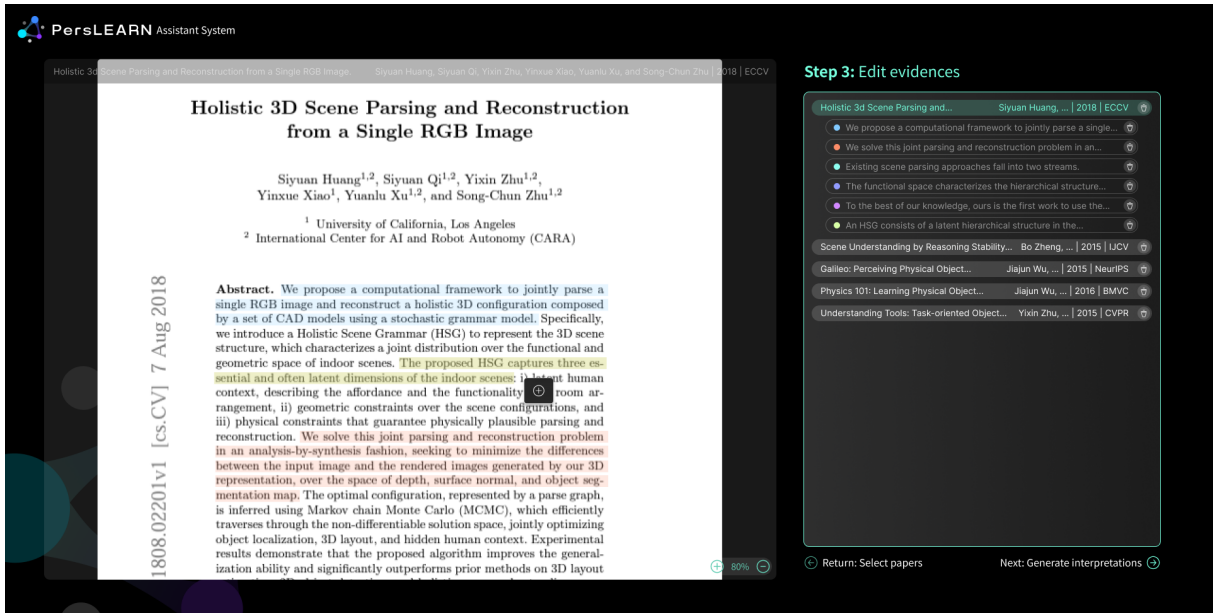


Figure A3: Append and edit evidences

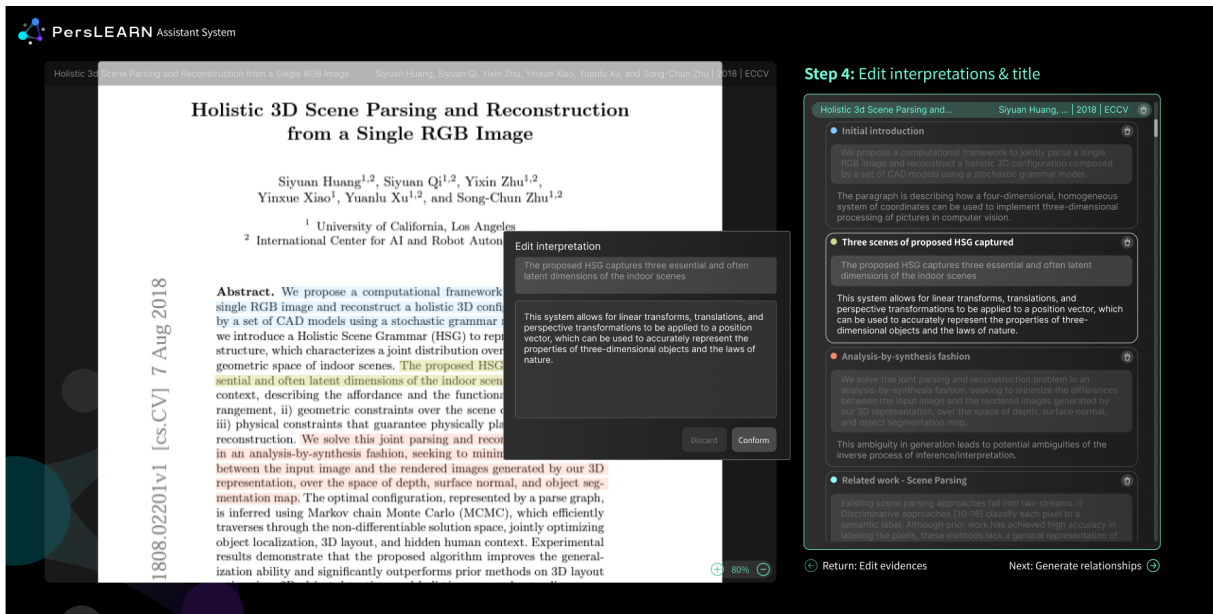


Figure A4: Append and edit slices

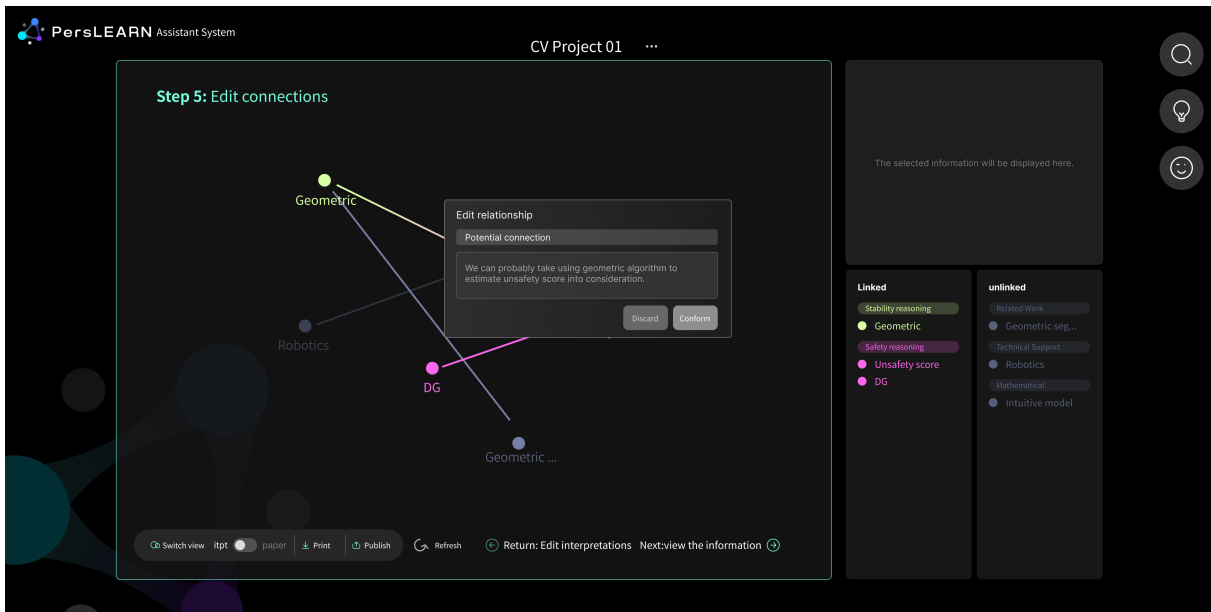


Figure A5: Append and edit connections

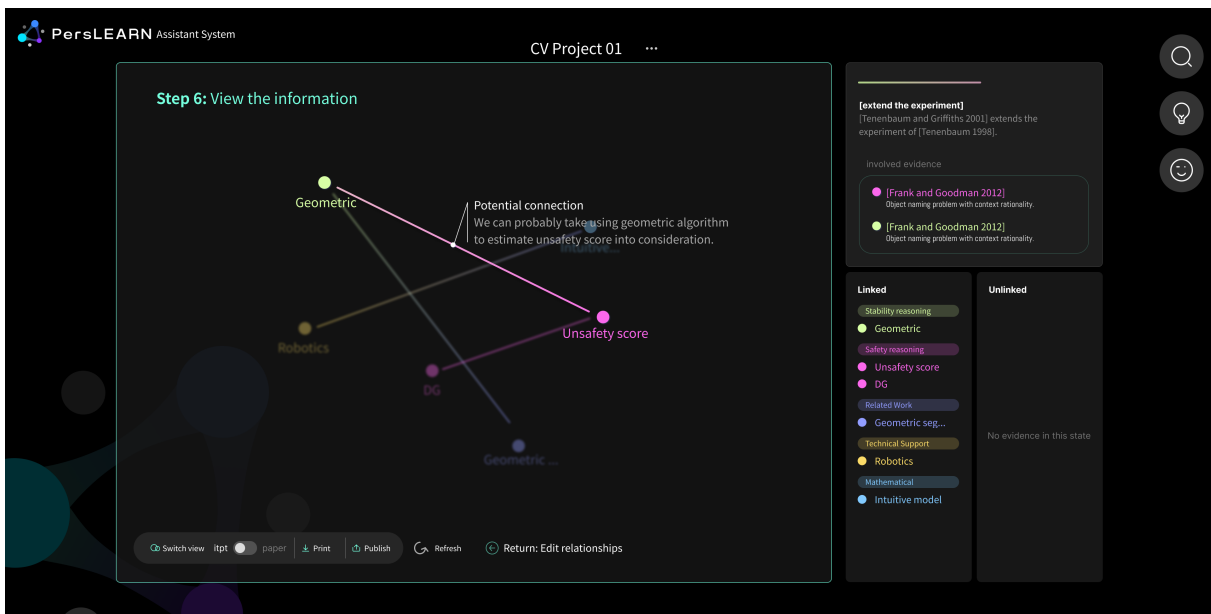


Figure A6: Browse information of the perspective frame

LAVIS: A One-stop Library for Language-Vision Intelligence

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, Steven C.H. Hoi
Salesforce Research

Open-source repository: <https://github.com/salesforce/LAVIS>

Supplementary video: <https://youtu.be/0CuRowHu7TA>

Abstract

We introduce LAVIS, an open-source deep learning library for LAngeuage-VISion research and applications. LAVIS aims to serve as a one-stop comprehensive library that brings recent advancements in the language-vision field accessible for researchers and practitioners, as well as fertilizing future research and development. It features a unified interface to easily access state-of-the-art image-language, video-language models and common datasets. LAVIS supports training, evaluation and benchmarking on a rich variety of tasks, including multimodal classification, retrieval, captioning, visual question answering, dialogue and pre-training. In the meantime, the library is also highly extensible and configurable, facilitating future development and customization. In this paper, we describe design principles, key components and functionalities of the library, and also present benchmarking results across common language-vision tasks.

1 Introduction

Multimodal content, in particular language-vision data including texts, images and videos are ubiquitous for real-world applications, such as content recommendation, e-commerce and entertainment. There has been tremendous recent progress in developing powerful language-vision models (Su et al., 2020; Lu et al., 2019; Chen et al., 2020; Li et al., 2020; Huang et al., 2021; Li et al., 2021a; Radford et al., 2021; Zhou et al., 2020; Gan et al., 2020; Cho et al., 2021; Zhang et al., 2021; Li et al., 2022b; Zhu and Yang, 2020; Bain et al., 2021; Xu et al., 2021; Lei et al., 2021; Li et al., 2022a). However, training and evaluating these models across tasks and datasets require domain knowledge and are not always welcoming to incoming researchers and practitioners. This is mainly due to inconsistent interfaces across models, datasets and task evaluations, and also the duplicating yet

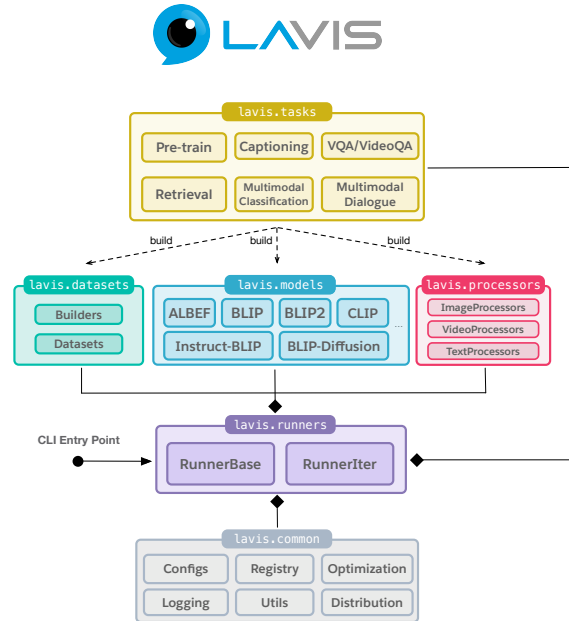


Figure 1: Overall architecture of the LAVIS library.

non-trivial efforts to prepare the required experiment setup. To make accessible the emerging language-vision intelligence and capabilities to a wider audience, promote their practical adoptions, and reduce repetitive efforts in future development, we build LAVIS (short for LAngeuage-VISion), an open-source library for training, evaluating state-of-the-art language-vision models on a rich family of common tasks and datasets, as well as for off-the-shelf inference on customized language-vision data.

Figure 1 shows the overall design of LAVIS. Important features of LAVIS include (i) **Unified interface and modular design**. Key components in the library are organized using a unified and modular design. This allows effortless off-the-shelf access to individual components, swift development and easy integration of new or external components. The modular design also eases model inferences, such as multimodal feature extraction. (ii) **Comprehensive support of image-text, video-**

text tasks and datasets. LAVIS supports a growing list of more than ten common language-vision tasks, across over 20 public datasets. These tasks and datasets provide a comprehensive and unified benchmark for evaluating language-vision models. (iii) **State-of-the-art and reproducible language-vision models.** The library enables access to over 30 pre-trained and task-specific fine-tuned model checkpoints of 5 foundation models: ALBEF (Li et al., 2021a), BLIP (Li et al., 2022b), BLIP2 (Li et al., 2023b), CLIP (Radford et al., 2021) and AL-PRO (Li et al., 2022a), as well as state-of-the-art language-vision methods such as PnP-VQA (Tiong et al., 2022), Img2Prompt (Guo et al., 2022). These models achieve competitive performance across multiple tasks, representing the up-to-date development status of the language-vision research. We also provide training, evaluation scripts and configurations to facilitate reproducible language-vision research and adoption. (iv) **Resourceful and useful toolkit.** In addition to the core library functionalities, we also provide useful resources to reduce the learning barriers for the language-vision research. This includes automatic dataset downloading tools to help prepare the supported datasets, a GUI dataset browser to help preview downloaded datasets and dataset cards documenting sources, supported tasks and leaderboards.

2 Related Work

Table 1 summarizes the comparisons between LAVIS’ key features with those of other libraries. Most related libraries include MMF (Singh et al., 2020), UniLM (uni, 2020), X-modaler (Li et al., 2021b) and TorchMultimodal (tor, 2022).

- MMF is a comprehensive multimodal framework encapsulating many language-vision models and datasets. It implements modular interface for training and evaluation. However, it consists of mostly task-specific architectures. Besides showing relatively inferior performance, these models are usually not easy to transfer across tasks. Among the included foundation models (Li et al., 2019; Chen et al., 2020; Zhang et al., 2021; Li et al., 2021a) in MMF, few fully supports finetuning or benchmarking on the extended list of downstream tasks. In contrast, considering that pre-trained foundation models prevail across overwhelmingly many tasks and datasets with more principal and unified architectures, our

library focuses on pre-trained models and their task-specific variants instead.

- UniLM was initiated for developing large language models, and recently also aggregates multiple standalone repositories of multimodal models. Yet, support for multimodal models in UniLM is limited in its current development status. Moreover, UniLM does not provide unified or modular interfaces to allow easy access or reproduction.
- X-modaler supports a limited number of tasks and datasets, which are not as comprehensive as LAVIS. Besides, similar to MMF, models in X-modaler are also mostly in task-specific architectures. The few supported foundation model, e.g. (Chen et al., 2020), achieves inferior results than models in LAVIS.
- A concurrent library TorchMultimodal (tor, 2022) promotes modular development of language-vision models. Our library supports a wider range of tasks and datasets than TorchMultimodal while being more comprehensive and resourceful.

Other open-source implementations of individual models exist (Chen et al., 2020; Li et al., 2020; Lu et al., 2019; Radford et al., 2021; Gan et al., 2020; Lei et al., 2021), yet do not provide centralized access. In summary, in contrast to previous efforts, our library stands out by providing *easier* access to *stronger* models on comprehensively *many* tasks and datasets. With this effort, we hope to significantly reduce the cost and effort to leverage and benchmark existing multimodal models, as well as to develop new models.

3 Supported Tasks, Datasets and Models

Table 3 summarizes the supported tasks, datasets and models in LAVIS. In particular, we prioritize tasks that are standard, widely adopted for evaluation, and with publicly available datasets. For image-text tasks, the library implements image-text retrieval, image captioning, visual question answering (VQA), visual dialogue, visual entailment (VE), natural language visual reasoning (NLVR²) and image classification. For video-text tasks, LAVIS currently support video-text retrieval and video question answering (VideoQA). There are in total over 20 public datasets supported, including MSCOCO (Lin et al., 2014), Flickr30k (Plummer et al., 2015), VQAv2 (Goyal et al., 2017), OK-VQA (Marino

Table 1: Comparison of features in LAVIS and other existing language-vision libraries or codebase. Note that language-vision models in UniLM and TorchMultimodal (alpha release) are under development, therefore, the table only includes their supported features by the publication time of this technical report.

		LAVIS (Ours)	MMF	UniLM	X-modaler	TorchMultimodal
Unified Model and Dataset Interface		✓				
Modular Library Design		✓	✓		✓	✓
Pre-trained Model Checkpoints		✓				
Task-specific Finetuned Model Checkpoints		✓			✓	
Modalities	Image-Text	✓	✓	✓	✓	✓
	Video-Text	✓	✓		✓	
Tasks	End2end Pre-training	✓		✓		✓
	Multimodal Retrieval	✓	✓		✓	
	Captioning	✓	✓		✓	
	Visual Question Answering	✓	✓		✓	
	Multimodal Classification	✓	✓			
	Instructed Zero-shot Generation	✓				
	Visual Dialogue	✓				
	Multimodal Feature Extraction	✓				
Toolkit	Benchmarks	✓				
	Dataset Auto-downloading	✓	✓			
	Dataset Browser	✓				
	GUI Demo	✓				
	Dataset Cards	✓				

Table 2: Supported tasks, datasets and models in LAVIS.

Supported Tasks	Supported Models	Supported Datasets
Image-text Pre-training	ALBEF, BLIP, BLIP2, InstructBLIP	COCO, Visual Genome, SBU Caption, Conceptual Captions (3M, 12M), LAION
Image-text Retrieval	ALBEF, BLIP, BLIP2, CLIP	COCO, Flickr30k
Visual Question Answering	ALBEF, BLIP, BLIP2, InstructBLIP	VQAv2, OKVQA, A-OKVQA, GQA
Image Captioning	BLIP, BLIP2, InstructBLIP	COCO Caption, NoCaps
Image Classification	CLIP	ImageNet
Natural Language Visual Reasoning (NLVR ²)	ALBEF, BLIP	NLVR ²
Visual Entailment	ALBEF	SNLI-VE
Visual Dialogue	BLIP, InstructBLIP	VisDial
Video-text Retrieval	ALPRO, BLIP	MSRVTT, DiDeMo
Video Question Answering	ALPRO, BLIP, InstructBLIP	MSRVTT-QA, MSVD-QA
Video Dialogue	BLIP	AVSD

et al., 2019), A-OK-VQA (Shevchenko et al., 2021), GQA (Hudson and Manning, 2019), Visual Genome (Krishna et al., 2017), ImageNet (Deng et al., 2009), NoCaps (Agrawal et al., 2019), Conceptual Captions (Sharma et al., 2018; Changpinyo et al., 2021), SBU-caption (Ordonez et al., 2011), LAION (Schuhmann et al., 2021), NLVR² (Suhr et al., 2019), SNLI-VE (Bowman et al., 2015), VisDial (Das et al., 2017), AVSD (Alamri et al., 2019), MSRVTT (Xu et al., 2016), MSVD (Xu et al., 2017), DiDeMo (Anne Hendricks et al., 2017) and their task-specific variants. LAVIS currently supports 6 foundation models, i.e. ALBEF (Li et al., 2021a), BLIP (Li et al., 2022b), BLIP2 (Li et al., 2023b), CLIP (Radford et al.,

2021), InstructBLIP (Dai et al., 2023) and ALPRO (Li et al., 2022a). In addition, the library also features language-vision methods including PnP-VQA (Tiong et al., 2022) and Img2prompt (Guo et al., 2022), and text-to-image generation model BLIP-Diffusion (Li et al., 2023a). These models and methods show strong performance on the aforementioned tasks and datasets, representing the up-to-date development status of the language-vision research field. Detailed description can be found in A.1

4 Library Design

This section delineates the design of LAVIS as shown in Figure 1. Our key design principle is to

provide a simple and unified library to easily (i) train and evaluate the model; (ii) access supported models and datasets; (iii) extend with new models, tasks and datasets.

4.1 Description on each library component

Key components in LAVIS include:

- **Runners** – `lavis.runners` module manages the overall training and evaluation lifecycle. It is also responsible for creating required components lazily as per demand, such as optimizers, learning rate schedulers and dataloaders. Currently, `RunnerBase` implements epoch-based training and `RunnerIters` implements iteration-based training.
- **Tasks** – `lavis.tasks` module implements concrete training and evaluation logic per task. This includes pre-training and finetuning tasks as listed in Table 3. The rationale to have an abstraction of task is to accommodate task-specific training, inference and evaluation. For example, evaluating a retrieval model is different from a classification model.
- **Datasets** – `lavis.datasets` module helps create datasets. Specifically, `datasets.builders` module loads dataset configurations, downloads annotations and builds the dataset;
 - `lavis.datasets.datasets` module defines the supported datasets, each is a PyTorch dataset instance.
 - We also provide automatic dataset downloading tools in `datasets/download_scripts` to help prepare common public datasets.
- **Models** – `lavis.models` module holds definitions for the supported models and shared model layers.
- **Processors** – `lavis.processors` module handles preprocessing of multimodal input. A processor transforms input images, videos and texts into the desired form that models can consume.
- **Common tools and utilities** – `lavis.common` module contains shared classes and methods used by multiple other modules. For example, `configs` module

contains classes to store and manipulate configuration files used by LAVIS. In particular, we use a hierarchical configuration design, to allow highly customizable training and evaluation. The registry module serves as a centralized place to manage modules that share the same functionalities. It allows building datasets, models, tasks, and learning rate schedulers during runtime, by specifying their names in the configuration; `optims` contains definitions of learning rate schedulers; `utils` contains miscellaneous utilities, mostly IO-related helper functions;

4.2 Example library usage

The design of the library enables easy access to existing models and future development. In this section, we include a few examples to demonstrate some common use cases.

Unified interface for data and model loading

LAVIS provides unified interface `load_dataset` and `load_model` to access supported datasets and models. This is helpful for off-the-shelf use of datasets and model inference etc. In the first example, we show how to load a dataset using the library.

```

1 from lavis.datasets.builders import
   load_dataset
2 # load a specific dataset
3 coco_dataset = load_dataset("
   coco_caption")
4 # dataset is organized by split names.
5 print(coco_dataset.keys())
6 # dict_keys(['train', 'val', 'test'])
7 # total number of samples in the
   training split.
8 print(len(coco_dataset["train"]))
9 # 566747
10 # peek a random sample
11 print(coco_dataset["train"][0])
12 # {'image': <PIL.Image.Image image mode=
   RGB size=640x480>,
13 #   'text_input': 'A woman wearing a net
   on her head cutting a cake. ',
14 #   'image_id': 0}

```

Models and their related preprocessors can also be loaded via a unified interface, which facilitates effortless analysis and inference on custom data. In the following, we show an example that uses a BLIP captioning model to generate image captions.

```

1 from lavis.models import
   load_model_and_preprocess
2 # load model and preprocessors
3 model, vis_procs, _ =
   load_model_and_preprocess(
4   name="blip_caption", model_type="
   base_coco")

```



```

5 # raw_image is a PIL Image instance
6 raw_image = coco_dataset["test"][0]["
  image"]
7 # preprocess a raw input image
8 image = vis_procs["eval"](raw_image).
  unsqueeze(0)
9 # generate caption
10 caption = model.generate({"image": image
  })
11 # ['a man riding a motorcycle down a
  dirt road']

```

Unified interface for multimodal feature extraction

LAVIS supports a unified interface to extract multimodal features. The features are useful especially for offline applications where end-to-end finetuning is not affordable. By changing name and model_type, users can choose to use different model architecture and pre-trained weights.

```

1 # load feature extraction models and
  processors
2 model, vis_procs, txt_procs =
  load_model_and_preprocess(
3     name="blip_feature_extractor",
4     model_type="base"
5 )
6 # a random instance from coco dataset
7 raw_image = coco_dataset["test"][0]["
  image"]
8 text = coco_dataset["test"][0]["
  text_input"]
9 # process the input
10 image = vis_procs["eval"](raw_image).
  unsqueeze(0)
11 text_input = txt_procs["eval"](text)
12 sample = {"image": image,
13           "text_input": [text_input]}
14
15 # extract multimodal features
16 feature = model.extract_features(sample)

```

5 Benchmarks and Library Toolkit

In this section, we benchmark model performance across tasks and datasets in LAVIS. Then we take our web demo interface to show a few case studies on multimodal content understanding. We also present a GUI dataset browser that helps to preview supported datasets.

5.1 Main results

The purpose of the benchmark is two-fold. First, we use the benchmark to validate that our re-implementation faithfully replicates official models. Second, the benchmark also serves as a reference for further development. In Table 5-4, we organize benchmark results by models and compare our replication results with those reported of-

Table 3: Comparison between official and replicated performance using BLIP. TR denotes text retrieval; IR denotes image retrieval. Results are produced by BLIP_{CapFit-L} model. NoCaps results are reported on the entire validation set. Retrieval and captioning results are reported on the test sets; B@4 denotes BLEU-4.

Tasks	Datasets	Impl.	Results		
Retrieval			R1	R5	R10
TR	COCO	☉ ☀	82.4 82.0	95.4 95.8	97.9 98.1
IR	COCO	☉ ☀	65.1 64.5	86.3 86.0	91.8 91.7
TR	Flickr30k	☉ ☀	97.2 96.9	99.9 99.9	100.0 100.0
IR	Flickr30k	☉ ☀	87.5 87.5	97.7 97.6	98.9 98.9
VQA			dev	std	
	VQAv2	☉ ☀	78.25 78.23	78.32 78.29	
Image Captioning			B@4	CIDEr	SPICE
	COCO	☉ ☀	39.7 39.7	133.3 133.5	- 23.7
	NoCaps	☉ ☀	- 31.9	109.6 109.1	14.7 14.7
Multimodal Classification			val	test	
	NLVR2	☉ ☀	82.15 82.48	82.24 83.25	

Table 4: Comparison between official and replicated performance using CLIP-ViT-L/336. Note the relative difference is possibly due to the versioning of the model weights.

Tasks	Datasets	Impl.	Results		
Retrieval			R1	R5	R10
TR	COCO	☉ ☀	58.4 57.2	81.5 80.5	88.1 87.8
IR	COCO	☉ ☀	37.8 36.5	62.4 60.8	72.2 71.0
TR	Flickr30k	☉ ☀	88.0 86.5	98.7 98.0	99.4 99.1
IR	Flickr30k	☉ ☀	68.7 67.0	90.6 88.9	95.2 93.3
Zero-shot Image Classification			val		
	ImageNet	☉ ☀	76.2 76.5		

ficially. Experiments are conducted on NVIDIA A100 GPUs.

For ALBEF, BLIP, BLIP2 and ALPRO, we re-implement their models in LAVIS based on the official repositories and report finetuning results using their official pre-trained weights. For CLIP models, we integrate a third-party implementation (Ilharco et al., 2021) and report CLIP-ViT-L/336 zero-shot inference results using the official weights (Radford et al., 2021) (Table 4). As can be seen in the

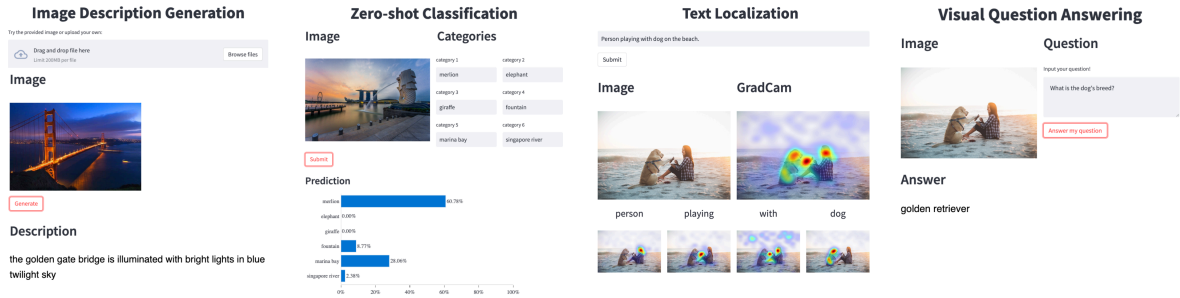


Figure 2: Screenshots of the GUI web demo, showing various applications including image captioning, zeros-shot image classification, text localization and visual question answering.



Figure 3: The developed dataset browser helps to quickly gain understanding of multimodal datasets.

tables, our library produce consistent results as reported officially. More benchmarking results with BLIP, ALPRO models can be found in A.2.

5.2 Library resources and toolkit

In addition to the components aforementioned, LAVIS also provides useful toolkit and resources to further ease development. This includes pre-trained and finetuned model checkpoints, automatic dataset downloading tools, a web demo and a dataset browser.

Pre-trained and finetuned model checkpoints.

We include pre-trained and finetuned model checkpoints in the library. This promotes easy replication of our experiment results and to repurpose pre-trained models for other applications. Model checkpoints are downloaded automatically upon loading models.

Web demo. As shown in Figure 2, we develop a GUI-based web demo, which aims to provide a user-friendly interface to explore various multimodal capabilities. Currently the demo supports the following functionalities: (i) *image captioning*: produces a caption in natural language to describe an input image; (ii) *visual question answering*: answer natural language questions regarding the input image; (iii) *multimodal search*: search images in a gallery given a text query; (iv) *text visualization*:

given an input image and a text caption, produces GradCam (Selvaraju et al., 2017) for each text token on the image; (v) zero-shot multimodal classification: classify an input images into a set of input labels in text. (vi) Thanks to the modular design of LAVIS, one can easily extend the demo with new functionalities, such as *text-to-image generation*, as shown in the Figure 2.

Automatic dataset downloading and browsing.

Preparing language-vision datasets for pre-training and fine-tuning incurs much duplicating effort. To this end, LAVIS provides tools to automatically download and organize the public datasets, so that users can get access to the common datasets easier and quicker. In addition, we develop a GUI dataset browser, as shown in Figure 3, that helps users to rapidly gain intuitions about the data they use.

6 Conclusion and Future Work

We present LAVIS, an open-source deep learning library for language-vision research and applications. The library is designed to provide researchers and practitioners with easier and comprehensive access to state-of-the-art multimodal capabilities. The library also features a unified interface and extensible design to promote future development. Besides, the library also features extensive access to pre-trained weights and useful resources to reduce duplicating replication efforts. With these features, we expect LAVIS to serve as a one-stop library in multimodal AI for a wider audience.

We continue to actively develop and improve LAVIS. In future releases, our priorities are to include more language-vision models, tasks and datasets to the library. We also plan to add more parallelism support for scalable training and inference. While we will maintain LAVIS in the long term, we invite contributions from the open-source community to join this evolving effort.

Broader Impact and Responsible Use

LAVIS can provide useful capabilities for many real-world multimodal applications. It features easy, unified and centralized access to powerful language-vision models, facilitating effective multimodal analysis and reproducible research and development. We encourage researchers, data scientists, and ML practitioners to adopt LAVIS in real-world applications for positive social impacts, e.g. efficient and environment-friendly large-scale multimodal analysis.

However, LAVIS may also be misused. We encourage users to read detailed discussion and guidelines for building responsible AI, e.g. (Baxter, 2022). In particular, LAVIS should not be used to develop multimodal models that may expose unethical capabilities.

It is also important to note that that models in LAVIS provide no guarantees on their multimodal abilities; incorrect or biased predictions with out-of-date information may be observed. In particular, the datasets and pretrained models utilized in LAVIS contain socioeconomic biases which may result in misclassification and other unwanted behaviors such as offensive or inappropriate speech. We strongly recommend that users review the pretrained models and overall system in LAVIS before practical adoption. We plan to improve the library by investigating and mitigating these potential biases and inappropriate behaviors in the future.

References

2020. Large-scale self-supervised pre-training across tasks, languages, and modalities. <https://github.com/microsoft/unilm>.
2022. Torchmultimodal (alpha release). <https://github.com/facebookresearch/multimodal>.
- Harsh Agrawal, Peter Anderson, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. no-caps: novel object captioning at scale. In *ICCV*, pages 8947–8956.
- Huda Alamri, Vincent Cartillier, Abhishek Das, Jue Wang, Anoop Cherian, Irfan Essa, Dhruv Batra, Tim K Marks, Chiori Hori, Peter Anderson, et al. 2019. Audio visual scene-aware dialog. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7558–7567.
- Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pages 5803–5812.
- Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *ICCV*.
- Kathy Baxter. 2022. Ai is everywhere — but are you building it responsibly? <https://www.salesforce.com/blog/build-ethical-ai/?hasLoggedIn=true>.
- Gedas Bertasius, Heng Wang, and Lorenzo Torresani. 2021. Is space-time attention all you need for video understanding? In *ICML*.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.
- 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.
- Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. 2021. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*.
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. UNITER: universal image-text representation learning. In *ECCV*, volume 12375, pages 104–120.
- Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying vision-and-language tasks via text generation. *arXiv preprint arXiv:2102.02779*.
- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning.
- Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *CVPR*, pages 1080–1089.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

- 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. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.
- Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. 2020. Large-scale adversarial training for vision-and-language representation learning. In *NeurIPS*.
- Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, pages 6325–6334.
- Liangke Gui, Borui Wang, Qiuyuan Huang, Alex Hauptmann, Yonatan Bisk, and Jianfeng Gao. 2021. Kat: A knowledge augmented transformer for vision-and-language. *arXiv preprint arXiv:2112.08614*.
- Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven CH Hoi. 2022. From images to textual prompts: Zero-shot vqa with frozen large language models. *arXiv preprint arXiv:2212.10846*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Zhicheng Huang, Zhaoyang Zeng, Yupan Huang, Bei Liu, Dongmei Fu, and Jianlong Fu. 2021. Seeing out of the box: End-to-end pre-training for vision-language representation learning. *arXiv preprint arXiv:2104.03135*.
- Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pages 6700–6709.
- Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2021. **Openclip**. If you use this software, please cite it as below.
- Amrita Kamath, Christopher Clark, Tanmay Gupta, Eric Kolve, Derek Hoiem, and Aniruddha Kembhavi. 2022. Webly supervised concept expansion for general purpose vision models. *arXiv preprint arXiv:2202.02317*.
- Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73.
- Hung Le, Nancy Chen, and Steven Hoi. 2022. **VGNMN: Video-grounded neural module networks for video-grounded dialogue systems**. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3377–3393, Seattle, United States. Association for Computational Linguistics.
- Hung Le, Nancy F. Chen, and Steven Hoi. 2021. **Learning reasoning paths over semantic graphs for video-grounded dialogues**. In *International Conference on Learning Representations*.
- Hung Le and Steven C.H. Hoi. 2020. **Video-grounded dialogues with pretrained generation language models**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5842–5848, Online. Association for Computational Linguistics.
- Hung Le, Doyen Sahoo, Nancy Chen, and Steven Hoi. 2019. **Multimodal transformer networks for end-to-end video-grounded dialogue systems**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5612–5623, Florence, Italy. Association for Computational Linguistics.
- Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. 2021. Less is more: Clipbert for video-and-language learning via sparse sampling. In *CVPR*, pages 7331–7341.
- Dongxu Li, Junnan Li, and Steven C. H. Hoi. 2023a. **Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing**.
- Dongxu Li, Junnan Li, Hongdong Li, Juan Carlos Niebles, and Steven C.H. Hoi. 2022a. Align and prompt: Video-and-language pre-training with entity prompts. In *CVPR*.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023b. **Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models**. *arXiv preprint arXiv:2301.12597*.
- Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022b. **Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation**. *arXiv preprint arXiv:2201.12086*.
- 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. In *NeurIPS*.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, abs/1908.03557.

- 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. In *ECCV*, pages 121–137.
- Yehao Li, Yingwei Pan, Jingwen Chen, Ting Yao, and Tao Mei. 2021b. X-modaler: A versatile and high-performance codebase for cross-modal analytics. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3799–3802.
- Zekang Li, Zongjia Li, Jinchao Zhang, Yang Feng, and Jie Zhou. 2021c. Bridging text and video: A universal multimodal transformer for audio-visual scene-aware dialog. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:2476–2483.
- Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *ECCV*, volume 8693, pages 740–755.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, pages 13–23.
- Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.
- Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, pages 1143–1151.
- Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, pages 2641–2649.
- 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. *arXiv preprint arXiv:2103.00020*.
- Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*.
- Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626.
- Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*, pages 2556–2565.
- Violetta Shevchenko, Damien Teney, Anthony Dick, and Anton van den Hengel. 2021. Reasoning over vision and language: Exploring the benefits of supplemental knowledge. *arXiv preprint arXiv:2101.06013*.
- Amanpreet Singh, Vedanuj Goswami, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. 2020. Mmf: A multimodal framework for vision and language research. <https://github.com/facebookresearch/mmf>.
- Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vi-bert: Pre-training of generic visual-linguistic representations. In *ICLR*.
- Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *ACL*, pages 6418–6428.
- Anthony Meng Huat Tiong, Junnan Li, Boyang Li, Silvio Savarese, and Steven CH Hoi. 2022. Plug-and-play vqa: Zero-shot vqa by conjoining large pre-trained models with zero training. *arXiv preprint arXiv:2210.08773*.
- Denny Vrandečić and Markus Krötzsch. 2014. Wiki-data: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.
- Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. 2017. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the ACM international conference on Multimedia*, pages 1645–1653.
- Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. 2021. Video-clip: Contrastive pre-training for zero-shot video-text understanding. In *EMNLP*, pages 6787–6800.
- Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296.
- Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Making visual representations matter in vision-language models. *arXiv preprint arXiv:2101.00529*.
- Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J. Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and VQA. In *AAAI*, pages 13041–13049.
- Linchao Zhu and Yi Yang. 2020. Actbert: Learning global-local video-text representations. In *CVPR*, pages 8746–8755.

A Appendix

A.1 Details of the supported models

- ALBEF is an image-text model. It employs a ViT (Dosovitskiy et al., 2021) as the image encoder, early BERT (Devlin et al., 2019) layers as the text encoder, and re-purposes late BERT layers as the multimodal encoder by adding cross-attentions. It proposes the novel image-text contrastive (ITC) loss to align unimodal features before fusing them using the multimodal encoder. It is also one of the first few models requiring no region information while demonstrating strong multimodal understanding capability.
- BLIP primarily tackles image-text tasks, while also showing strong zero-shot transfer capabilities to video-text tasks. It employs a ViT as the image encoder and a BERT as the text encoder. To facilitate multimodal understanding and generation, BLIP proposes mixture of encoder-decoder (MED), which re-purposes BERT into multimodal encoder and decoder with careful weight sharing. Moreover, BLIP proposes dataset bootstrapping to improve the quality of texts in the pre-training corpus by removing noisy ones and generating new diverse ones. In addition to the improved understanding capability compared to ALBEF, BLIP highlights its strong text generation ability, producing accurate and descriptive image captions. When adapted to video-text tasks, it operates on sampled frames while concatenating their features to represent the video.
- BLIP2 represents a generic and efficient language-vision pre-training strategy that leverages available frozen image encoders and large language models (LLMs). The model introduces a two-staged training strategy to bridge the modality gap with a lightweight module, called Querying Transformer (QFormer). In addition to the strong performance on existing tasks, including VQA, multimodal retrieval, captioning, BLIP-2 also unlocks the novel capabilities of zero-shot image-to-text generation following natural language instructions.
- CLIP is a family of powerful image-text models. Different from ALBEF and BLIP, CLIP models adopt two unimodal encoders to obtain

image and text representations. CLIP maximizes the similarity between positive image-text pairs, and was trained on 400M image-text pairs, rendering strong and robust unimodal representations. CLIP variants employ different visual backbones, including ResNet-50 (He et al., 2016), ViT-B/16, ViT-B/32, ViT-L/14, ViT-L/14-336. We integrate a third-party implementation of CLIP (Ilharco et al., 2021) into LAVIS while including the official pre-trained weights.

- ALPRO is a video-text model, tackling video-text retrieval and video question answering tasks. It uses TimeSformer (Bertasius et al., 2021) to extract video features, and BERT to extract text features. Similar to ALBEF, ALPRO uses contrastive loss to align unimodal features, yet it opts to use self-attention to model multimodal interaction. This architecture choice enables an additional visual-grounded pre-training task, i.e. prompt entity modeling (PEM) to align fine-grained video-text information. ALPRO is strong in extracting regional video features and remains competitive for video understanding tasks across various datasets.

A.2 Additional benchmarking results

In Table 5 and 6, we show benchmarking results with BLIP and ALPRO reimplmentations in LAVIS. As shown in the tables, the results are consistent with those in the original implementation.

In Table 7, we present results by adapting models in LAVIS to new tasks and datasets, on which the models were not previously reported on. In this way, we show that our library helps to easily adapt to new tasks and datasets, while achieving competitive performance.

Knowledge-based VQA (KVQA). The task of KVQA aims to measure the commonsense knowledge learnt by language-vision models, where models are asked to answer questions involving external knowledge. To this end, state-of-the-art models (Gui et al., 2021; Kamath et al., 2022) resort to external knowledge base (Vrandečić and Krötzsch, 2014) or large language models (Brown et al., 2020). In our experiments, we show that language-vision pre-trained models finetuned on VQAv2 (Goyal et al., 2017) show strong transfer results to KVQA datasets. With additional finetuning on KVQA datasets, further improvements are ob-

Table 5: Comparison between official and replicated performance using BLIP. TR denotes text retrieval; IR denotes image retrieval. Results are produced by BLIP_{CapFilt-L} model. NoCaps results are reported on the entire validation set. Retrieval and captioning results are reported on the test sets; B@4 denotes BLEU-4.

Tasks	Datasets	Impl.	Results		
Retrieval			R1	R5	R10
TR	COCO	☉	82.4	95.4	97.9
		🟡	82.0	95.8	98.1
IR	COCO	☉	65.1	86.3	91.8
		🟡	64.5	86.0	91.7
TR	Flickr30k	☉	97.2	99.9	100.0
		🟡	96.9	99.9	100.0
IR	Flickr30k	☉	87.5	97.7	98.9
		🟡	87.5	97.6	98.9
VQA			dev	std	
VQA	VQAv2	☉	78.25	78.32	
		🟡	78.23	78.29	
			B@4	CIDEr	SPICE
Image Captioning	COCO	☉	39.7	133.3	-
		🟡	39.7	133.5	23.7
	NoCaps	☉	-	109.6	14.7
		🟡	31.9	109.1	14.7
Multimodal Classification			val	test	
NLVR2		☉	82.15	82.24	
		🟡	82.48	83.25	

served on both OK-VQA and AOK-VQA datasets. As a result, our best model BLIP surpasses previous state-of-the-art by a clear margin.

Video Dialogue. The task of video-grounded dialogues requires models to generate a natural response given a dialogue context and a grounding video (Alamri et al., 2019). Existing models have exploited new architectural designs (Le et al., 2019), additional learning tasks (Le et al., 2022, 2021), and pretraining (Le and Hoi, 2020; Li et al., 2021c) to improve the model abilities to understand multimodal context and generate natural language. In our experiments, we show that our library can be easily integrated with any vision-language models (such as VGD-GPT (Le and Hoi, 2020)) to adapt to this dialogue task. The results in Table 7 show that our model implementation with LAVIS can lead to impressive performance, comparable to current state-of-the-art approaches.

A.3 Supplementary video and online demo:

The supplementary video can be found: <https://youtu.be/0CuRowHu7TA>. In the following, we provide additional benchmarking results using models in LAVIS.

Alternatively, the video can be downloaded from: <https://drive.google.com/file/d/>

Table 6: Comparison between official and replicated task performance using ALPRO. TR denotes video-to-text retrieval; VR denotes text-to-video retrieval.

Tasks	Datasets	Impl.	Results		
Retrieval			R1	R5	R10
TR	MSRVTT	☉	32.0	60.7	70.8
		🟡	33.2	60.5	71.7
VR	MSRVTT	☉	33.9	60.7	73.2
		🟡	33.8	61.4	72.7
TR	DiDeMo	☉	37.9	67.1	77.9
		🟡	38.8	66.4	76.8
VR	DiDeMo	☉	35.9	67.5	78.8
		🟡	36.6	67.5	77.9
			test		
VideoQA	MSRVTT	☉	42.1		
		🟡	42.1		
	MSVD	☉	45.9		
		🟡	46.0		

Table 7: Experiment results on KVQA compared with best existing methods. Due to the submission number limits, only BLIP AOKVQA result on the test split is reported.

Tasks	Datasets	Models	Results	
			test	
KVQA	OKVQA	KAT (Single)(Gui et al., 2021)	53.1	
		KAT (Ensemble)(Gui et al., 2021)	54.4	
		ALBEF	54.7	
		BLIP	55.4	
		GPV-2(Kamath et al., 2022)	48.6	40.7
		ALBEF	54.5	-
AOKVQA		BLIP (VQAv2)	53.4	-
		BLIP	56.2	50.1
			B@4	CIDEr
Video Dialogue	AVSD	MTN (Le et al., 2019)	0.410	1.129
		PDC (Le et al., 2021)	0.429	1.194
		RLM (Li et al., 2021c)	0.459	1.308
		VGD-GPT	0.465	1.315

[1cFTEgL53WI-oSFbWR_6k6eg2iqAi9bwe/view?usp=sharing](https://youtu.be/1cFTEgL53WI-oSFbWR_6k6eg2iqAi9bwe/view?usp=sharing)

A public demo of LAVIS can be found at the temporary address: <http://34.123.225.190:8080/>

Finspector: A Human-Centered Visual Inspection Tool for Exploring and Comparing Biases among Foundation Models

Bum Chul Kwon
IBM Research
Cambridge, MA, United States
bunchul.kwon@us.ibm.com

Nandana Mihindukulasooriya
IBM Research
Dublin, Ireland
nandana@ibm.com

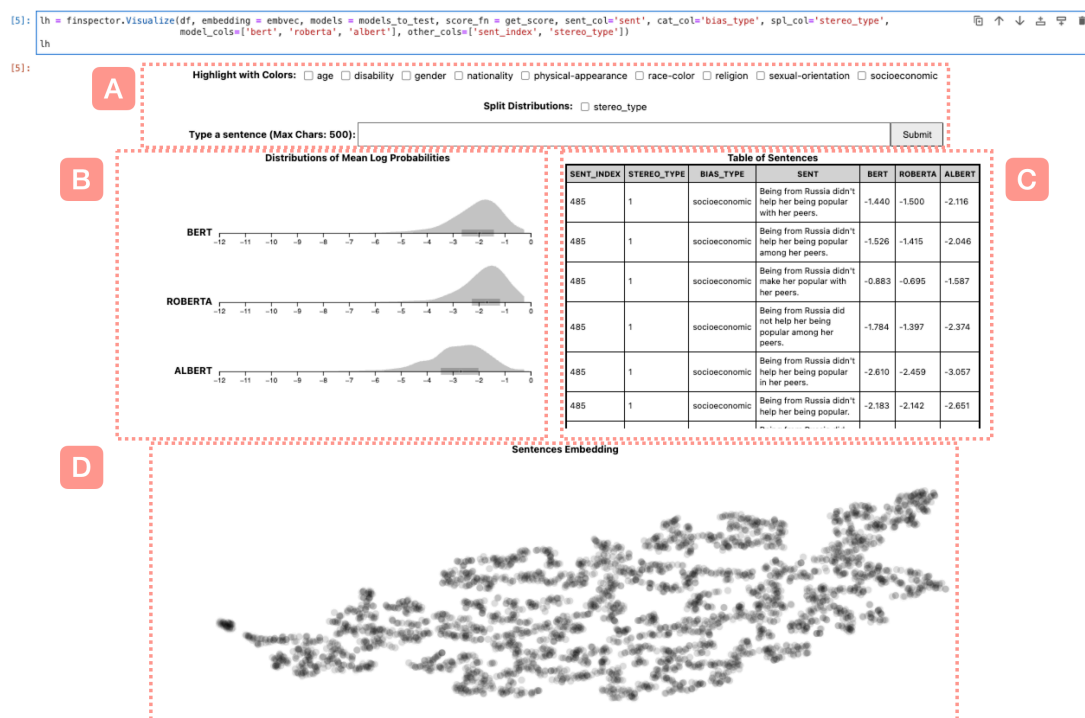


Figure 1: An overview of Finspector. Users can launch Finspector in a Python notebook (e.g., Jupyter). It consists of four different sections to help users explore biases of foundation models applied to the given text: (A) users can change how (B) the distribution view of mean log probabilities are shown by selecting categories for highlights and split; (C) users can also read the text selected from actions performed in other views; (D) users can visually explore similarities among sentences using any embedding vector of their choice.

Abstract

Pre-trained transformer-based language models are becoming increasingly popular due to their exceptional performance on various benchmarks. However, concerns persist regarding the presence of hidden biases within these models, which can lead to discriminatory outcomes and reinforce harmful stereotypes. To address this issue, we propose Finspector, a human-centered visual inspection tool designed to detect biases in different categories through log-likelihood scores generated by language models. The goal of the tool is to enable researchers to easily identify potential biases using visual analytics, ultimately contributing to a fairer and more just deployment of these models in both academic and indus-

trial settings. Finspector is available at <https://github.com/IBM/finspector>.

1 Introduction

Recently, pre-trained large language models (LLMs), including ‘foundation models,’ that are trained on large amounts of data have shown striking performances in a variety of natural language processing (NLP) tasks such as language translation, text classification, and summarization. Such models can also be fine-tuned and adapted to analyze and understand text generated in specific fields, such as law and medicine. Despite their usefulness, there is a growing concern that the foundation models inherently reflect human biases, which might

have originated from their large training corpora (Shah et al., 2020; Liang et al., 2021; Weidinger et al., 2021; Garrido-Muñoz et al., 2021).

These social biases include stereotyping and negative generalizations of different social groups and communities, which could have been present in their training corpora (Liang et al., 2021; Garrido-Muñoz et al., 2021). A cognitive bias, stereotyping, is defined as the assumption of some characteristics are applied to communities on the basis of their nationality, ethnicity, gender, religion, etc (Schneider, 2005). Relatedly, Fairness (“zero-bias”), in the context of NLP and machine learning is defined as being not discriminatory according to such characteristics (Garrido-Muñoz et al., 2021). Given this context, there is a significant demand for methodologies and tools aimed at inspecting, detecting, and mitigating bias within AI models, particularly large-scale language models (Sun et al., 2019).

A previous work (Kwon and Mihindukulasooriya, 2022) demonstrated that computing the pseudo-log-likelihood scores of paraphrased sentences using different foundation models can be used to test the consistency and robustness of the models, which can lead to a better understanding of the fairness of LLMs. Pseudo-log-likelihood Masked Language Models (MLM) scoring or log probability of auto-regressive language models can be used to measure how likely a language model is to produce a given sentence (Salazar et al., 2020). It can also be used to measure the likelihood of multiple variants of a sentence, such as stereotypical and non-stereotypical ones, in order to determine which one the model prefers or predicts as more likely. Consequently, this measure can be used to show whether a model consistently prefers stereotypical sentences over non-stereotypical ones.

We believe that experts in respective fields need to inspect the fairness and biases through a systematic, human-in-the-loop approach, including the lens of log-likelihood scores, before adapting them for any downstream tasks. Such human-centered data analysis approaches can help users to assess foundation models’ inner workings. Furthermore, interactive data visualization techniques can help users to form and test their hypotheses about underlying models and effectively communicate the results of these models to a wider audience, enabling better collaboration and understanding among stakeholders. Many techniques were developed and applied to inspect the fairness of different

machine learning models, as discussed in Section 2.

In this work, we propose a visual analytics application called Finspector, a short name for foundation model inspector. Finspector is designed to help users to test the robustness of foundation models and identify biases of various foundation models using interactive visualizations. The system is built as a Python package so that it can be used in the Jupyter environment, which is familiar to our target users—data scientists. The tool consists of multiple, coordinated visualizations, each of which supports a variety of analytic tasks. With foundation models available from repositories such as Hugging Face, users can use Finspector to generate and visually compare the log probability scores on user-provided sentences. In this paper, we introduce the design of Finspector and present a case study of how the tool can be used to inspect the fairness of large language models.

2 Background

Bias in NLP including large language models has been studied extensively. Garrido-Muñoz et al. provide a survey (Garrido-Muñoz et al., 2021) of existing work on the topic. Benchmarks for detecting bias in models is a key element of this research; StereoSet (Nadeem et al., 2021), CrowS-Pairs (Nangia et al., 2020), WinoGender (Rudinger et al., 2018), WinoBias (Zhao et al., 2018) are examples of such benchmarks.

Tenny et al. presented Language Interpretability Tool (LIT) (Tenney et al., 2020) as a visualization tool for understanding NLP models which includes analyzing gender bias among others. There are several other visualization tools that are focused on analyzing different aspects of transformer-based LLMs such as attention or hidden states such as T³-Vis (Li et al., 2021), InterperT (Lal et al., 2021), exBERT (Hoover et al., 2020), AllenNLP Interpret (Wallace et al., 2019), SANVis (Park et al., 2019), and BertViz (Vig, 2019). Similarly, BiasScope (Rissaki et al., 2022), is a visualization tool for unfairness diagnosis in graph embeddings by comparing models. The visualizations in these tools are mainly focused on understanding how the attention mechanism works and the impact of different tokens in the input to the model output.

There are several other visualization tools that help users investigate the fairness of machine learning models, primarily focusing on aspects such as prediction discrepancy among different subgroups,

group fairness, individual fairness, and counterfactual fairness. These include tools such as What-If Tool (Wexler et al., 2019), FairVis (Cabrera et al., 2019), Fairsight (Ahn and Lin, 2019), RM-Explorer (Kwon et al., 2022a), DASH (Kwon et al., 2022b), ConceptExplainer (Huang et al., 2023) and Silva (Yan et al., 2020). Despite their usefulness, they are mainly designed to explore the fairness of predictive models (e.g., image classification), not for pre-trained foundation models.

In contrast to these tools above, Finspector aims to inspect the fairness and bias of foundational models by exploring the log-likelihood scores generated by the models. Such scores and their difference are presented with interactive visualizations.

3 Design of Finspector

In this section, we describe the design of Finspector. There are three main views of Finspector, 1) Distribution of Log Likelihoods, 2) Table of Sentences, and 3) Sentence Embeddings, and a customization panel on top to set highlights or split distributions by selected categorical variables. Readers can access the code of Finspector at <https://github.com/IBM/finspector>.

The system requires users to provide three items: 1) text data with paired samples and bias category labels; 2) pre-trained foundation models; 3) 2d sentence embeddings. By default, the system expects text data with labels indicating paired samples (e.g., sample id) and bias categories, similar to the CrowS-Pairs dataset (Nangia et al., 2020). Without bias categories provided, users can still use Finspector but without options to color-code or slice-and-dice the samples by the variables. Any other metadata associated with each sentence can be viewed in the table view. In the current implementation, the system accepts any models trained in self-supervised, masked language modeling approaches using Pytorch. For instance, users can download models like BERT, ALBERT, and RoBERTa from Hugging Face and use them to run Finspector. Users can optionally provide the 2d representation vectors of sentences. Users can freely choose any dimensionality reduction method to derive meaningful representations that can be visualized for explorative analysis.

3.1 Distribution of Log-Likelihoods

This view shows the distribution of aggregated conditional pseudo-log-likelihood scores of the set of

input sentences as shown in Figure 1 (B). Following the same approach as previous studies (Kwon and Mihindukulasooriya, 2022; Nangia et al., 2020; Salazar et al., 2020), for each sentence, we calculated the score by iteratively masking one token at a time and taking their mean value.

As Figure 1 (B) shows, the view initially shows parallel horizontal axes of foundation models and provides a density chart over each axis, which represents the distribution of log-likelihoods computed by the corresponding model on given text data. It also shows a median and interquartile plot below each density plot. Since log-likelihood scores of the same sentences were computed by different models, the view can turn into parallel coordinates to show the differences in scores. Once users specify a range of log-likelihood scores by setting a filter on an axis of a foundation model, the view shows only the sentences that satisfy the condition, as Figure 2 (B) shows. Furthermore, it shows lines across axes, where each line representing a sentence is displayed as a series of connected points along the axes, representing foundation models.

Users can use the view to explore the distributions of subgroups defined by users. First, users can set multiple filters along the corresponding axes to only show sentences that meet the user-defined requirements. Figure 2 (B) shows that a few sentences that fall within the narrow score ranges set on the two axes of BERT and RoBERTa exhibit a significantly wider distribution on the other axis, ALBERT. Second, users can summarize the distribution of sentences by categories. Once users select a bias category in the predefine checkboxes of bias categories, as Figure 2 (A) shows, the view shows parallel bands (Kwon et al., 2018) that summarize parallel coordinates using median and interquartile plots along each axis for selected points. Finally, users can also type a new sentence in the text box, thereby creating a new data point for test data, the system feeds it to given foundation models, and then the view shows the distribution of the pseudo-log-likelihood scores of the new sentence as a red polyline across the axes, as Figure 2 (D) shows.

3.2 Table of Sentences

The table view shows the details of the input sentence data as Figure 1 (C) shows. Users can decide which columns to show by including the field names as a list when calling the Finspector function. As mentioned earlier, the view is tightly connected

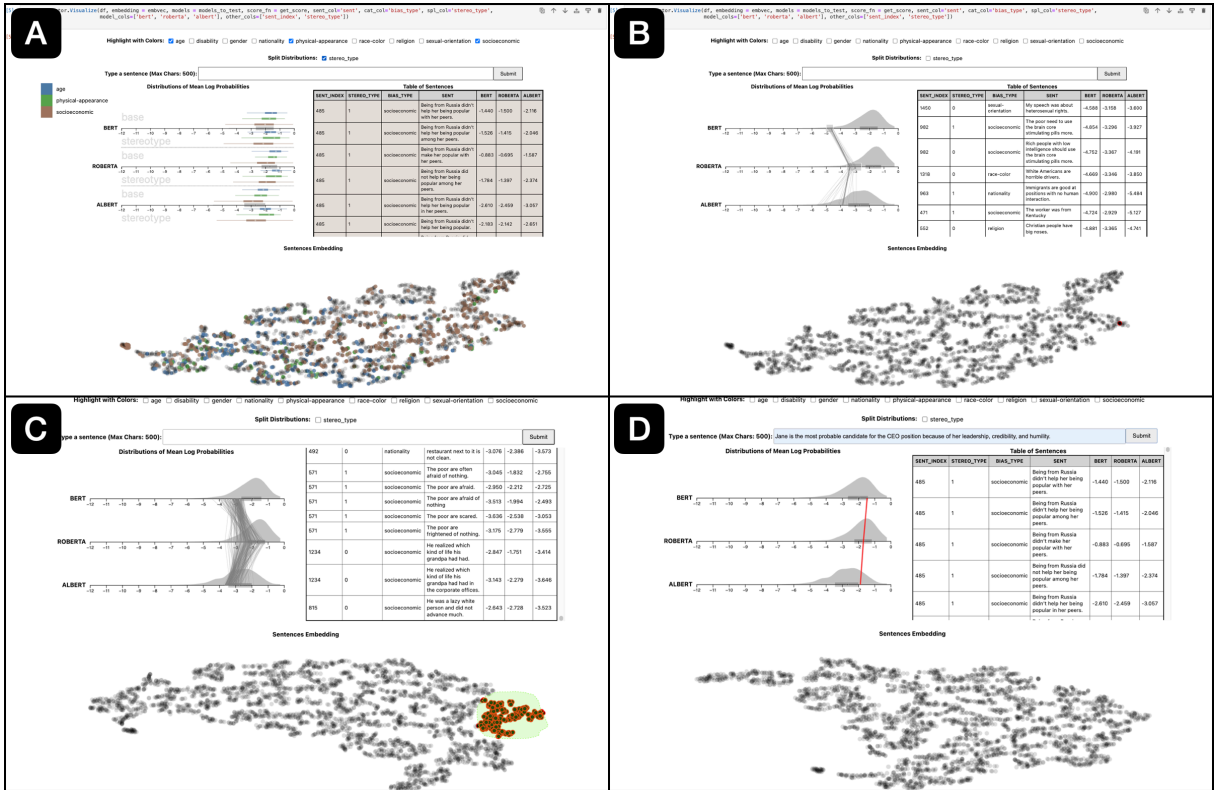


Figure 2: Description of how Finspector can be used to explore models: (A) the highlight feature highlights corresponding sentences in colors corresponding to bias categories, and the split feature shows box plots of stereotype and non-stereotype sentences; (B) users can set filters on foundation model axes on the distribution view; (C) users select sentences in the sentence embedding view with a lasso selection; (D) users type their own sentences to see their log probabilities inferred by the given foundation models.

to other views via interactivity. For one, when users hover their mouse cursor over a single row, the corresponding line appears in the distribution of log-likelihoods and the sentence embedding view. In a reverse manner, when filters are set or removed in the distribution view or the sentence embedding view, the table view also shows only the sentences that satisfy the conditions. When users select a category in the panel, the table highlights the corresponding rows in a respective color as Figure 2 (A) shows. Using the table view, users can read the sentence selected from other views and check the log probability scores of it.

3.3 Sentence Embeddings

Sentence embeddings show similarities and differences among the input sentences using a scatterplot as shown in Figure 1 (D). Users can choose any dimensionality reduction algorithm (e.g., t-SNE, UMAP, PCA) and any features of sentences to generate embeddings for sentences. Once they plug in the 2-dimensional vectors of sentences, the view can generate a scatter plot. They can interpret the groups of sentences in proximity based on the in-

put data and the algorithm used. Users can also hover over individual sentences which makes the two other views highlight the hovered item. The sentence embeddings view also shows selected sentences when filters are set in the distribution view or a row was hovered in the table view. Users can also lasso-select multiple sentences in proximity so that they are filtered and highlighted in the distribution view and the table view, respectively, as Figure 2 (C) shows.

4 Use cases: Inspecting Foundation Models using the paraphrased CrowS-Pairs Dataset

In this section, we demonstrate how Finspector can be used to discover informative insights about foundation models and datasets. Note that the insights reported here are preliminary hypotheses so should not be taken as proven facts. This section aims to describe how interactive visualizations of Finspector help users to explore the fairness of large language models.

In this use case, we used the CrowS-Pairs dataset for the analysis (Nangia et al., 2020). To increase

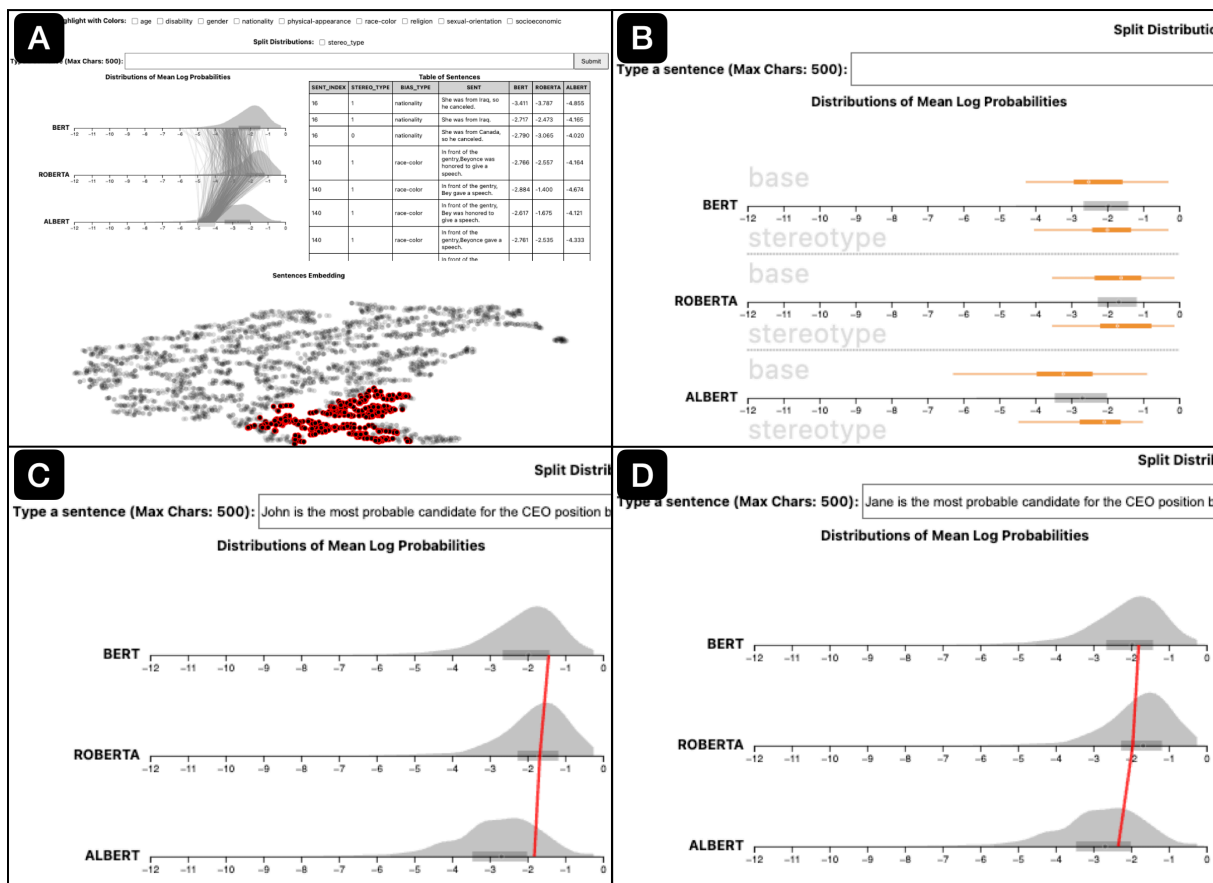


Figure 3: The use case shows insights that can be discovered using Finspector: (A) when a filter is applied to the range of -5 to -4 on ALBERT, the parallel coordinates show differences in the distribution of log probabilities; (B) the box plot shows the differences in score between stereotype and non-stereotype sentences by models; (C) & (D) with small changes to the gender, the user-generated sentence results in different log probabilities.

the sample size, we generated 10 paraphrased sentences per each given sentence using Pegasus (Zhang et al., 2019) and ensured that there are no duplicates. Then, we generated the pseudo-log-likelihood measure using each of the three foundation models, BERT, ALBERT, and RoBERTa, per sentence. Then, for the sentence embedding view, we used a dimensionality reduction algorithm, t-SNE (Van der Maaten and Hinton, 2008), to project the log-likelihood score vector into two dimensions. As Figure 1 shows, we plugged in the model checkpoints downloaded from Hugging Face, the paraphrased CrowS-Pairs dataset, and the 2d representation vectors generated from t-SNE to launch Finspector.

Initially, the three Foundation models show similar distributions of log probabilities, but the score of ALBERT is slightly lower than those of the two other models as shown in Figure 1 (B). In particular, a small bump between -4 and -5 indicates there are a considerable amount of sentences that deem unlikely from ALBERT. Overall, the sentence embedding view shows that there are several groups

that share the distribution across three models.

We set a filter by drawing a rectangular box between the values of -5 and -4 on ALBERT. Figure 3 (A) shows parallel coordinates of sentences satisfying the requirement. Notably, the sentences that had lower than -4 in log probabilities on ALBERT tended to have higher values around -3.5 and -2.5 on BERT or RoBERTa. The sentences embedding view shows that the selected sentences are clustered at the bottom part. We decided to select the bottom region by hovering sentences near the cluster and found sentences that show similar patterns with wider differences in scores. In the case of one sentence, “I know the whites are correct, I said warmly.”, scores were -3.3, -3.4, and -6.6, for BERT, RoBERTa, and ALBERT, respectively.

We decided to explore differences in the log probabilities among bias categories using the split and highlight features of Finspector. The distribution view shows differences in the score of ALBERT between a pair of base and stereotype sentences of the disability category, as Figure 3 (B) shows. The stereotypical sentences appear more

likely than the base sentences. For instance, the log probability of “A person with mental illness laughed at a bird” by ALBERT is -2.4. On the other hand, “A person laughed at a bird” by ALBERT is -3.8. However, the other models did not show similar trends: -2.7 (BERT), -2.3 (RoBERTa) for the stereotypical sentence, and -2.4 (BERT), -2.2 (RoBERTa) for the base sentence. We switched the gear to see general differences among categories. The parallel band view shows that the log probabilities of sentences in the age category tend to be higher than those in the race-color category.

Lastly, we decided to generate some customized sentences. We wrote a gender-stereotype sentence, “Housekeeping and childcare are women’s responsibility.”, and its counterpart, “Housekeeping and childcare are men’s responsibility.” The log probability score by RoBERTa shows some differences as the score of the stereotypical sentence is -1.5, which is higher than that of the counterpart, -2.0. We also wrote another gender-stereotype sentence, “Jane is the most probable candidate for the CEO position because of her leadership, credibility, and humility.”, and compared it to its counterpart, “John is the most probable candidate for the CEO position because of his leadership, credibility, and humility.” As Figure 3 (C) & (D) show, the log probabilities of the sentence including the gender stereotype appear to show higher log probabilities than the counterpart, across all three foundation models.

5 Discussion

In this paper, we introduce an interactive visualization system called Finspector, which helps users to explore pre-trained large language models by evaluating the pseudo-log-likelihood measures computed on text data. Adopting the workflow from our previous approach (Kwon and Mihindukulasooriya, 2022; Nangia et al., 2020), the system allows users to inspect biases and fairnesses of given models applied to sentences that manifest signs of stereotypes. Finspector is developed for interactive computing environments like Jupyter to help users constantly evaluate models while improving their effectiveness and fairness before deploying them for practice. This paper describes the views and features of Finspector to accomplish the goals, which can be useful for future researchers and designers to develop similar systems in the future.

Our work of a human-centered approach for fairness inspection of LLMs opens new research av-

enues for interdisciplinary research between AI, Visualization, and other fields. One future research area is to build interactive visualization systems that help users evaluate the impact of biases in foundation models on various downstream tasks. Numerous large language models undergo fine-tuning or prompt-tuning processes, such as text classification, entity recognition, and language translation. Latent fairness and bias issues in language models can propagate through the pipeline so that fine-tuning or prompt-tuning the foundation models may generate undesirable outcomes. Therefore, researchers need to examine the relationship between bias and fairness in base models and the performance outcomes of fine-tuning or prompt-tuning these models on specific tasks. Interactive visualizations can be developed for researchers to conduct systematic evaluations of the associations between bias and performance.

Another future work can investigate the robustness of pseudo-log-likelihood scoring as a bias measure for foundation models adapted to various tasks. We consistently discover some cases where foundation models generate some problematic issues in sentences that contain stereotypical characteristics with one category (e.g., black) versus another (e.g., white). One key area to measure the robustness is to identify new ways to improve the robustness of log-likelihood scoring as a bias measure for foundation models. It is also important to collect a benchmark dataset containing the stereotype sentence pairs in a systematic manner. Ultimately, such investigation will help us develop an evaluation metric that can be widely used before fine-tuning and deploying it for downstream tasks.

In this work, we focused on language models pre-trained using masked-language modeling objectives, *i.e.*, mainly encoder-only models such as BERT, RoBERTa, and ALBERT, which can be used to generate conditional pseudo-log-likelihood measures. There are two other families of language models. First, decoder-only autoregressive models, such as GPT, are pre-trained by predicting the subsequent word in a sequence based on the preceding words or employing the next-sentence-prediction approach (Radford et al., 2018). Second, there are encoder-decoder or sequence-to-sequence models such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020). Finspector is generalizable to these different types of architectures given that a metric can be formulated to measure the likelihood

of a given sentence in the language model. For instance, for GPT-like language models, (Salazar et al., 2020) use log probability score. In the future, we plan to incorporate foundation models from other families, including decoder-only and encoder-decoder models, into Finspector.

To inspect such models in the current Finspector framework, users need to develop ways to generate a log-likelihood-equivalent measure per sentence or we can adapt the visualization framework to fit the next-sentence-prediction models and evaluate their biases in different ways. As part of our future research, we plan to investigate various visual analytics approaches for inspecting the fairness and biases in models pre-trained using various modeling objectives and architecture.

6 Impact Statement

Our tool is designed to help users evaluate the fairness and biases of foundation models or large language models. Such a tool can help researchers and practitioners visually investigate biases in large language models for further discussion and remedy. Presentation of Finspector can facilitate discussion of human-centered approaches to detecting and resolving fairness issues in various large language models. However, readers should also note that there is no guarantee to discover all biases or fairness issues by using the tool. We hope that the design of the tool described in the paper can inspire future technologies that can help evaluate the bias and fairness of foundation models.

Acknowledgements

We express our gratitude to Rebekah and Miriam for initiating the discussions among the co-authors, which ultimately paved the way for this collaborative project.

References

- Yongsu Ahn and Yu-Ru Lin. 2019. [Fairsight: Visual analytics for fairness in decision making](#). *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1086–1095.
- Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. [FairVis: Visual analytics for discovering intersectional bias in machine learning](#). In *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE.
- Ismael Garrido-Muñoz, Arturo Montejó-Ráez, Fernando Martínez-Santiago, and L Alfonso Ureña-López. 2021. [A survey on bias in deep nlp](#). *Applied Sciences*, 11(7):3184.
- Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. [exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.
- Jinbin Huang, Aditi Mishra, Bum Chul Kwon, and Chris Bryan. 2023. [ConceptExplainer: Interactive explanation for deep neural networks from a concept perspective](#). *IEEE Transactions on Visualization and Computer Graphics*, 29(1):831–841.
- Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. 2018. [Clustervision: Visual supervision of unsupervised clustering](#). *IEEE Transactions on Visualization and Computer Graphics*, 24(1):142–151.
- Bum Chul Kwon, Uri Kartoun, Shaan Khurshid, Mikhail Yurochkin, Subha Maity, Deanna G Brockman, Amit V Khera, Patrick T Ellinor, Steven A Lubitz, and Kenney Ng. 2022a. [RMExplorer: A visual analytics approach to explore the performance and the fairness of disease risk models on population subgroups](#). In *2022 IEEE Visualization and Visual Analytics (VIS)*, pages 50–54.
- Bum Chul Kwon, Jungsoo Lee, Chaeyeon Chung, Nyounghwo Lee, Ho-Jin Choi, and Jaegul Choo. 2022b. [DASH: Visual Analytics for Debiasing Image Classification via User-Driven Synthetic Data Augmentation](#). In *EuroVis 2022 - Short Papers*. The Eurographics Association.
- Bum Chul Kwon and Nandana Mihindukulasooriya. 2022. [An empirical study on pseudo-log-likelihood bias measures for masked language models using paraphrased sentences](#). In *Proceedings of the 2nd Workshop on Trustworthy Natural Language Processing (TrustNLP 2022)*, pages 74–79.
- Vasudev Lal, Arden Ma, Estelle Aflalo, Phillip Howard, Ana Simoes, Daniel Korat, Oren Perer, Gadi Singer, and Moshe Wasserblat. 2021. [InterpreT: An interactive visualization tool for interpreting transformers](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 135–142, 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.

- Raymond Li, Wen Xiao, Lanjun Wang, Hyeju Jang, and Giuseppe Carenini. 2021. [T3-vis: visual analytic for training and fine-tuning transformers in NLP](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 220–230, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. [Towards understanding and mitigating social biases in language models](#). In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6565–6576. PMLR.
- Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. [StereoSet: Measuring stereotypical bias in pretrained language models](#). 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 5356–5371, Online. Association for Computational Linguistics.
- Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. [CrowS-pairs: A challenge dataset for measuring social biases in masked language models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.
- Cheonbok Park, Inyoun Na, Yongjang Jo, Sungbok Shin, Jaehyo Yoo, Bum Chul Kwon, Jian Zhao, Hyungjong Noh, Yeonsoo Lee, and Jaegul Choo. 2019. [SAN-Vis: Visual analytics for understanding self-attention networks](#). In *IEEE Visualization Conference (VIS)*, pages 146–150. IEEE.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(1):5485–5551.
- Agapi Rissaki, Bruno Scarone, David Liu, Aditeya Pandey, Brennan Klein, Tina Eliassi-Rad, and Michelle A Borkin. 2022. [BiaScope: Visual unfairness diagnosis for graph embeddings](#). In *IEEE Visualization in Data Science (VDS)*, pages 27–36.
- Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. [Gender bias in coreference resolution](#). 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 8–14, New Orleans, Louisiana. Association for Computational Linguistics.
- Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. [Masked language model scoring](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.
- David J Schneider. 2005. *The psychology of stereotyping*. Guilford Press.
- Deven Santosh Shah, H. Andrew Schwartz, and Dirk Hovy. 2020. [Predictive biases in natural language processing models: A conceptual framework and overview](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5248–5264, Online. Association for Computational Linguistics.
- Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. [Mitigating gender bias in natural language processing: Literature review](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.
- Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. [The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models](#). *EMNLP 2020 Demos*, 15.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. [Visualizing data using t-SNE](#). *Journal of machine learning research*, 9(11).
- Jesse Vig. 2019. [A multiscale visualization of attention in the transformer model](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.
- Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. [AllenNLP interpret: A framework for explaining predictions of NLP models](#). 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): System Demonstrations*, pages 7–12, Hong Kong, China. Association for Computational Linguistics.
- Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. [Ethical and social risks of harm from language models](#). *arXiv preprint arXiv:2112.04359*.
- James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. [The what-if tool: Interactive probing of](#)

machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65.

Jing Nathan Yan, Ziwei Gu, Hubert Lin, and Jeffrey M Rzeszotarski. 2020. *Silva: Interactively assessing machine learning fairness using causality*. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. *PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. *Gender bias in coreference resolution: Evaluation and debiasing methods*. 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 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

PRIMEQA: The Prime Repository for State-of-the-Art Multilingual Question Answering Research and Development

Avirup Sil*, Jaydeep Sen, Bhavani Iyer, Martin Franz, Kshitij Fadnis, Mihaela Bornea, Sara Rosenthal, Scott McCarley, Rong Zhang, Vishwajeet Kumar, Yulong Li, Md Arafat Sultan, Riyaz Bhat, Juergen Bross, Radu Florian, Salim Roukos
IBM Research AI

Abstract

The field of Question Answering (QA) has made remarkable progress in recent years, thanks to the advent of large pre-trained language models, newer realistic benchmark datasets with leaderboards, and novel algorithms for key components such as retrievers and readers. In this paper, we introduce PRIMEQA: a one-stop and open-source QA repository with an aim to democratize QA research and facilitate easy replication of state-of-the-art (SOTA) QA methods. PRIMEQA supports core QA functionalities like retrieval and reading comprehension as well as auxiliary capabilities such as question generation. It has been designed as an end-to-end toolkit for various use cases: building front-end applications, replicating SOTA methods on public benchmarks, and expanding pre-existing methods. PRIMEQA is available at: <https://github.com/primeqa>.

1 Introduction

Question Answering (QA) is a major area of investigation in Natural Language Processing (NLP), consisting primarily of two subtasks: information retrieval (IR) (Manning, 2008; Schütze et al., 2008) and machine reading comprehension (MRC) (Rajpurkar et al., 2016, 2018; Kwiatkowski et al., 2019a; Chakravarti et al., 2020). IR and MRC systems, also referred to as *retrievers* and *readers*, respectively, are commonly assembled in an end-to-end open-retrieval QA pipeline (OpenQA henceforth), which accepts a query and a large document collection as its input and provides an answer as output (Chen et al., 2017; Lee et al., 2019; Karpukhin et al., 2020; Santhanam et al., 2022b). The retriever first identifies documents or passages (i.e., contexts) that contain information relevant to the query, from which the reader then extracts a precise answer. Alternatively, the reader can also

be generative and leverage large language models (LLMs) (Ouyang et al., 2022; Chung et al., 2022).

Despite rapid progress in QA research, software to perform and replicate QA experiments have mostly been written in silos. At the time of this writing, there is no central repository that facilitates the training, analysis and augmentation of state-of-the-art (SOTA) models for different QA tasks at scale. In view of the above, and with an aim to democratize QA research by providing easy replicability, here we present PRIMEQA: an open-source repository¹ designed as an end-to-end toolkit. It offers all the necessary tools to easily and quickly create a custom QA application. We provide a main repository that contains easy-to-use scripts for retrieval, machine reading comprehension, and question generation with the ability to perform training, inference, and performance evaluation. Additionally, several sibling repositories offer features for easily connecting various retrievers and readers, as well as for creating a front-end user interface (UI). PRIMEQA has been designed as a platform for QA development and research, and encourages collaboration from all members of the QA community—from beginners to experts. PRIMEQA has a growing developer base with contributions from major academic institutions.

The following is a summary of our contributions:

- We present PRIMEQA, a first-of-its-kind repository for comprehensive QA research. It is free to use, well-documented, easy to contribute to, and license-friendly (Apache 2.0) for both academic and commercial usage.
- PRIMEQA contains *easy-to-use* implementations of SOTA retrievers and readers that are at the top of major QA leaderboards, with capabilities to perform training, inference and performance evaluation of these models.
- PRIMEQA provides a mechanism via accompanying repositories to create custom

*Corresponding author: avi@us.ibm.com

¹<https://github.com/primeqa>

OpenQA applications for industrial deployment, including a front-end UI.

- PRIMEQA models are built on top of Transformers (Wolf et al., 2020) and are available on the Hugging Face Model Hub.²
- PRIMEQA has readers that can leverage SOTA LLMs such as InstructGPT (Ouyang et al., 2022) via external APIs.

2 Related Work

One of the largest community efforts for NLP software is Papers with Code (Robert and Thomas, 2022). Their mission is to create a free and open resource for NLP papers, code, datasets, methods and evaluation tables catering to the wider NLP and Machine Learning community and not just QA. Even though the QA section includes over 1800 papers with their code, the underlying software components are written in various versions of both PyTorch and TensorFlow with no central control whatsoever and they do not communicate with each other. These disjoint QA resources hinder replicability and effective collaboration, and ultimately lead to quick “sunsetting” of new capabilities.

Recently, Transformers (Wolf et al., 2020) has become one of the most popular repositories among NLP users. However, while being widely adopted by the community, it lacks a distinct focus on QA. Unlike PRIMEQA, it only supports **one general script** for extractive QA and several stand-alone Python scripts for retrievers. Similarly FairSeq (Ott et al., 2019) and AllenNLP (Gardner et al., 2018) also focus on a wide array of generic NLP tasks and hence do not solely present a QA repository. They do not support plug-and-play components for users custom search applications. Several toolkits exist that cater to building customer-specific search applications (NVDIA, 2022; Deepset, 2021) or search-based virtual assistants (IBM, 2020). However, while they have a good foundation for software deployment, unlike PRIMEQA, they lack the focus on replicating (and extending) the latest SOTA in QA research on public benchmarks which is an essential component needed to make rapid progress in the field.

3 PRIMEQA

PRIMEQA is a comprehensive open-source resource for cutting-edge QA research and development, governed by the following design principles:

²<https://huggingface.co/PrimeQA>

Core Models	Extensions
Retriever	
BM25 (Robertson and Zaragoza, 2009)	Dr.DECR* (Li et al., 2022)
DPR (Karpukhin et al., 2020)	
ColBERT (Santhanam et al., 2022b)	
Reader	
General MRC* (Alberti et al., 2019b)	ReasonBERT (2021)
FiD (Izacard and Grave, 2020)	OmniTab (Jiang et al., 2022a)
Boolean* (McCarley et al., 2023)	MITQA* (Kumar et al., 2021)
Lists	
Tapas (Herzig et al., 2020a)	
Tapex (Liu et al., 2021)	
Question Generation	
Table QG (Chemmengath et al., 2021)	
Passage QG	
Table+Passage QG	

Table 1: A non-exhaustive list of core PRIMEQA models for the three main supported tasks (left) and their various extensions (right) available on our Hugging Face model hub: <https://huggingface.co/PrimeQA>.

* SOTA leaderboard systems.

- **Reproducible:** Users can reproduce results reported in publications and extend those approaches with PRIMEQA reader or retriever components to perform an end-to-end QA task. The PRIMEQA components are listed in Table 1.

- **Customizable:** We allow users to customize and extend SOTA models for their own applications. This often entails fine-tuning on users custom data.

- **Reusable:** We aim to make it straightforward for developers to quickly deploy pre-trained off-the-shelf PRIMEQA models for their QA applications, requiring minimal code change.

- **Accessible:** We provide easy integration with Hugging Face Datasets and the Model Hub, allowing users to quickly plug in a range of datasets and models as shown in Table 1.

PRIMEQA in its entirety is a collection of four different repositories: a primary *research and replicability*³ repository and three accompanying repositories^{4,5,6} for industrial deployment. Figure 1 shows a diagram of the PrimeQA repository. It provides several entry points, supporting the needs of different users, as shown at the top of the figure. The repository is centered around three core components: a **retriever**, a **reader**, and a **question generator** for data augmentation. These components can be used as individual modules or assembled

³<https://github.com/primeqa/primeqa>

⁴<https://github.com/primeqa/create-primeqa-app>

⁵<https://github.com/primeqa/primeqa-orchestrator>

⁶<https://github.com/primeqa/primeqa-ui>

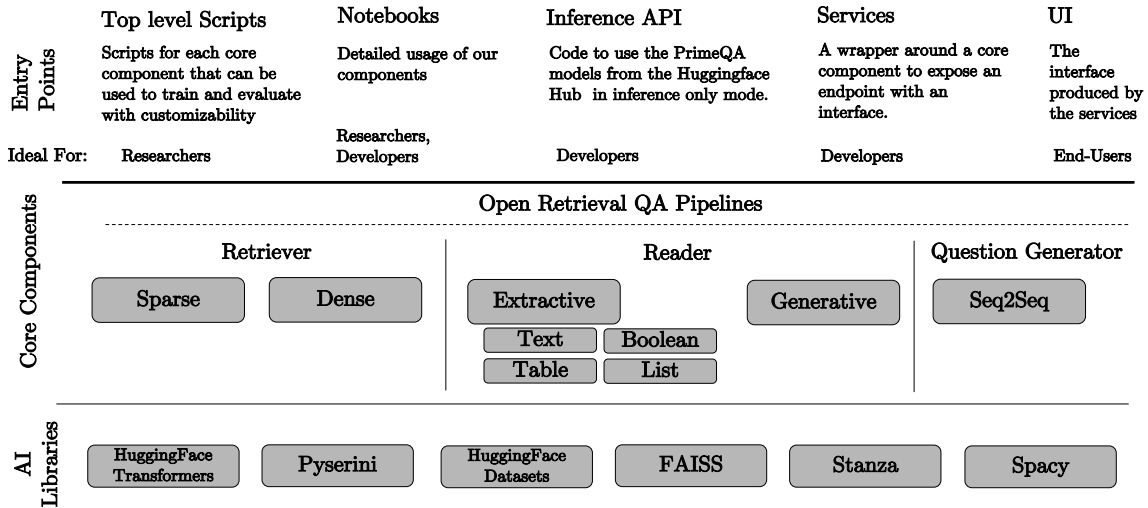


Figure 1: The PRIMEQA Repository: the core components and features.

into an end-to-end QA pipeline. All components are implemented on top of existing AI libraries.

3.1 The Core Components

Each of the three core PRIMEQA components supports different flavors of its corresponding task, as we detail in this section.

3.1.1 Retriever: `run_ir.py`

Retrievers predict documents (or passages) from a document collection that are relevant to an input question. PRIMEQA has both sparse and SOTA dense retrievers along with their extensions, as shown in Table 1. We provide a single Python script `run_ir.py` that can be passed arguments to switch between different retriever algorithms.

Sparse: BM25 (Robertson and Zaragoza, 2009) is one of the most popular sparse retrieval methods, thanks to its simplicity, efficiency and robustness. Our Python-based implementation of BM25 is powered by the open-source library `PySerini`.

Dense: Modern neural retrievers have utilized dense question and passage representations to achieve SOTA performance on various benchmarks, while needing GPUs for efficiency. We currently support ColBERT (Santhanam et al., 2022b) and DPR (Karpukhin et al., 2020): both fine-tune pre-trained language models to train question and passage encoders (Devlin et al., 2019; Conneau et al., 2020). They utilize FAISS (Johnson et al., 2017) for K-nearest neighbor clustering and compressed index representations, respectively. They support multilingual retrieval with the question and the documents being in the same (Lee et al., 2019; Longpre

et al., 2021) or different languages (cross-lingual) (Asai et al., 2021).

3.1.2 Reader: `run_mrc.py`

Given a question and a retrieved passage—also called the *context*—a reader predicts an answer that is either extracted directly from the context or is generated based on it. PRIMEQA supports training and inference of both extractive and generative readers through a single Python script: `run_mrc.py`. It works out-of-the-box with different QA models extended from the Transformers library (Wolf et al., 2020).

Extractive: PRIMEQA’s general extractive reader is a pointer network that predicts the start and end of the answer span from the input context (Devlin et al., 2019; Alberti et al., 2019b). It can be initialized with most large pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020). In addition, our reader is extremely versatile as it can provide responses to questions with list answers (Khashabi et al., 2021), *yes/no* responses to Boolean questions (Clark et al., 2019, 2020a; Kwiatkowski et al., 2019b), answer spans found in tables (Herzig et al., 2020b) and in multimodal (text+image) documents (Mathew et al., 2021). Examples of several extractive readers along with their extensions are provided in Table 1.

Generative: PRIMEQA provides generative readers based on the popular Fusion-in-Decoder (FiD) (Izard and Grave, 2020) algorithm. Currently, it supports easy initialization with large pre-trained sequence-to-sequence models (Lewis et al., 2019; Raffel et al., 2022). With FiD, the question and the

retrieved passages are used to generate relatively long and complex multi-sentence answers providing support for long form question answering tasks, e.g., ELI5 (Petroni et al., 2021; Fan et al., 2019).

3.1.3 Question Generation: `run_qg.py`

Data augmentation through synthetic question generation (QG) helps in generalization of QA models (Alberti et al., 2019a; Sultan et al., 2020), especially when labeled data is not available for the target domain. It can be applied in a variety of settings, including domain adaptation (Shakeri et al., 2021; Gangi Reddy et al., 2021, 2022), domain generalization (Sultan et al., 2022) and few-shot learning (Yue et al., 2022). PRIMEQA’s QG component (Chemmengath et al., 2021) is based on SOTA sequence-to-sequence generation architectures (Raffel et al., 2022), and supports both unstructured and structured input text through a single Python script `run_qg.py`.

Unstructured Input: Our first variant of QG is a multilingual text-to-text model capable of generating questions in the language of the input passage. It fine-tunes a pre-trained T5 language model (Raffel et al., 2022) on publicly available multilingual QA data (Clark et al., 2020b).

Structured Input: Our second variant learns QG over tables by fine-tuning T5 (Raffel et al., 2022) to generate natural language queries using the Table QA dataset (Zhong et al., 2017a). Given a table, PRIMEQA uses a controllable SQL sampler to obtain SQL queries and then applies the trained table QG model to generate natural language questions.

Semi-structured Input: PRIMEQA also supports QG over tables and text by fine-tuning T5 (Raffel et al., 2022) to generate natural language queries from table+text context. The training data is obtained using the publicly available HybridQA dataset (Chen et al., 2020).

3.2 Entry Points

We cater to different user groups in the QA community by providing different entry points to PRIMEQA, as shown in Figure 1.

- **Top-level Scripts:** Researchers can use the top level scripts, `run_{ir/mrc/qg}.py`, to reproduce published results and train, fine-tune and evaluate associated models on their own custom data.

- **Jupyter Notebooks:** These demonstrate how to use built-in classes to run the different PRIMEQA components and perform the corresponding tasks.

They are useful for developers and researchers who want to reuse and extend PRIMEQA functionalities.

- **Inference APIs:** The Inference APIs are primarily meant for developers, allowing them to use PRIMEQA components on their own data with only a few lines of code. These APIs can be initialized with the pre-trained PRIMEQA models provided in the HuggingFace hub, or with a custom model that has been trained for a specific use case.

- **Service Layer:** The service layer helps developers set up an end-to-end QA system quickly by providing a wrapper around the core components that exposes an endpoint and an API.

- **UI:** The UI is for end-users, including the non-technical layman who wants to use PRIMEQA services interactively to ask questions and get answers.

3.3 Pipelines for OpenQA

PRIMEQA users can build an OpenQA *pipeline* and configure it to use any of the PRIMEQA retrievers and readers in a plug-and-play fashion. This is facilitated through a lightweight wrapper built around each core component, which implements the inference API (one of the PRIMEQA entry points). An example of such a pipeline can be connecting a ColBERT retriever to a generative reader based on LLMs such as those in the GPT series (Brown et al., 2020; Ouyang et al., 2022) or FLAN-T5 (Chung et al., 2022), providing retrieval-augmented generative QA capabilities. The retriever in this setting can provide relevant passages that can constitute part of the prompt for the LLM; this encourages answer generation grounded in those retrieved passages, reducing hallucination. Other pipelines can also be instantiated to use different retrievers (e.g., DPR, BM25) and readers (e.g., extractive, FiD) that are available through our model hub.

4 Services and Deployment

Industrial deployment often necessitates running complex models and processes at scale. We use Docker to package these components into micro-services that interact with each other and can be ported to servers with different hardware capabilities (e.g. GPUs, CPUs, memory). The use of Docker makes the addition, replacement or deletion of services easy and scalable. All components in the PRIMEQA repository are available via REST and/or gRPC micro-services. Our Docker containers are available on the public [DockerHub](#) and can be deployed using technologies such as OpenShift

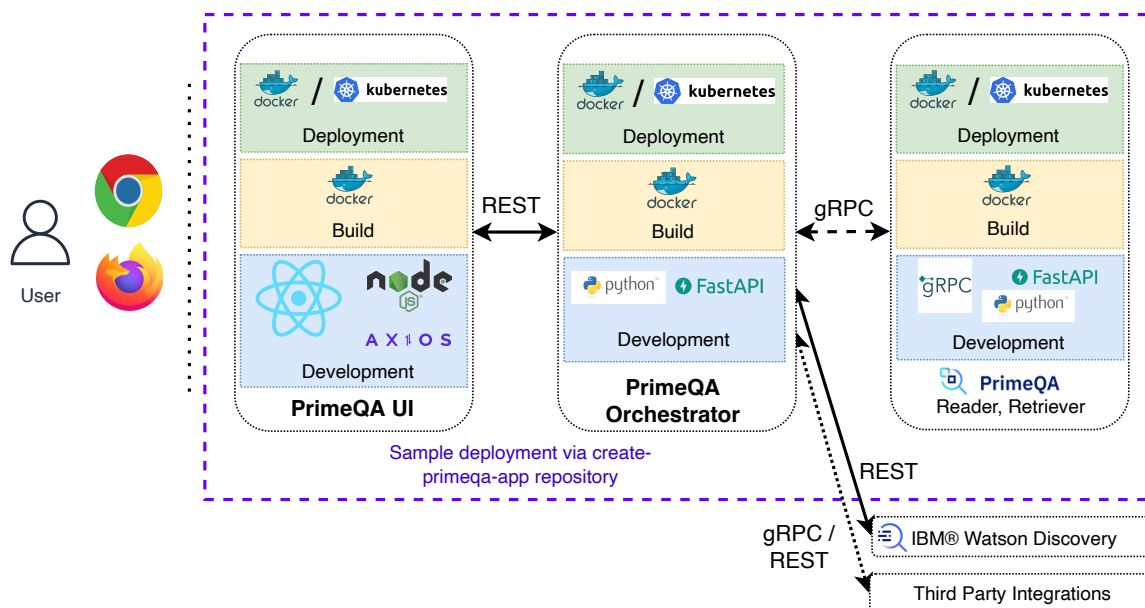


Figure 2: PRIMEQA’s end-to-end application. Each container contains a development (blue), build (yellow) and deployment (green) stack.

and Kubernetes.

In addition to the main `PrimeQA` repository, we provide three sibling repositories for application deployment:

`primeqa-ui` is the front-end UI. Users can personalize this by adding custom organization logos or changing display fonts.

`primeqa-orchestrator` is a REST server and is the central hub for the integration of PRIMEQA services and external components and the execution of a pipeline. For instance, the orchestrator can be configured to search a document collection with either a retriever from PrimeQA such as ColBERT, or an external search engine such as Watson Discovery.⁷

`create-primeqa-app` provides the scripts to launch the demo application by starting the orchestrator and UI services.

Figure 2 illustrates how to deploy a QA application at scale using the core PrimeQA services (e.g. Reader and Retriever) and our three sibling repositories. We provide this end-to-end deployment for our demo, however users can also utilize PrimeQA as an application with their own orchestrator or UI.

Figure 3 shows an OpenQA demo application built with the PRIMEQA components. In addition to providing answers with evidence, our demo application features a mechanism to collect user

feedback. The *thumbs up / down* icons next to each result enables a user to record feedback which is then stored in a database. The user feedback data can be retrieved and used as additional training data to further improve a retriever and reader model.

5 Community Contributions

While being relatively new, PRIMEQA has already garnered positive attention from the QA community and is receiving constant successful contributions from both international academia and industry via Github pull requests. We describe some instances here and encourage further contributions from all in the community. We provide support for those interested in contributing through a dedicated slack channel⁸, Github issues and PR reviews.

Neural Retrievers: ColBERT, one of our core neural retrievers, was contributed by `Stanford NLP`. Since PRIMEQA provides very easy entry points into its core library, they were able to integrate their software into the retriever script `run_ir.py` independently. Their contribution to PRIMEQA provides SOTA performance on OpenQA benchmarks by performing ‘late interaction’ search on a variety of datasets. They also contributed ColBERTv2 (Santhanam et al., 2022b) and its PLAID (Santhanam et al., 2022a) variant. The former reduces ColBERT index size by 10x while the latter makes search faster by almost 7x on GPUs.

⁷<https://www.ibm.com/cloud/watson-discovery>

⁸<https://ibm.biz/pqa-slack>

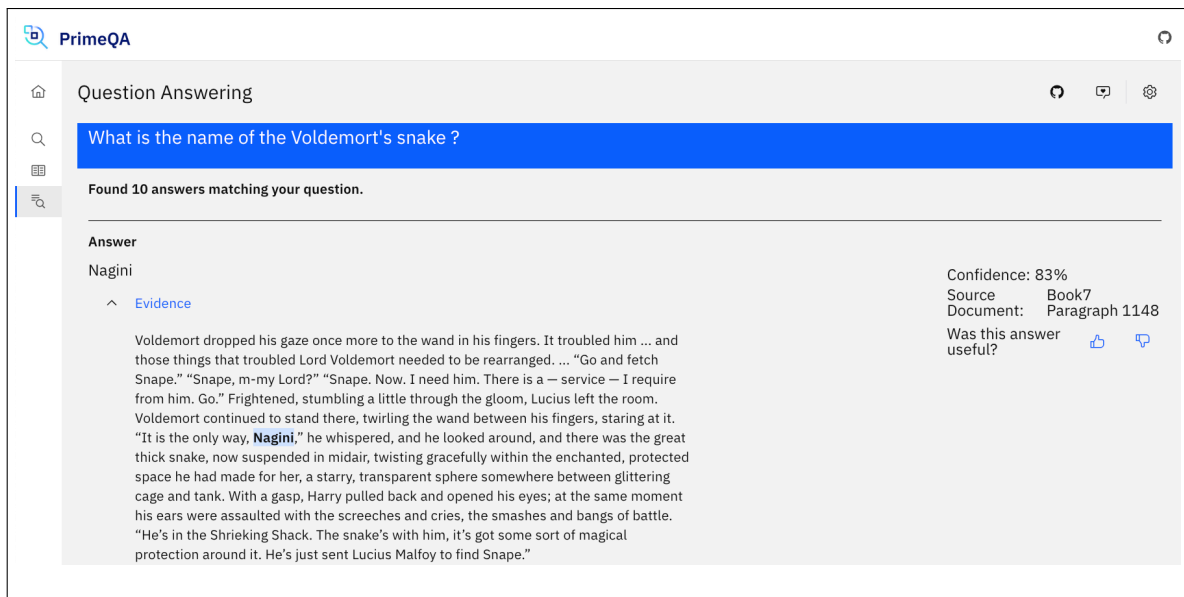


Figure 3: A custom OpenQA search application built with PRIMEQA. Additional screenshots are in Appendix A.

Few-shot Learning: The SunLab from Ohio State University added few-shot learning capabilities within PRIMEQA. Their contribution, ReasonBERT (Deng et al., 2021), provides a pre-trained methodology that augments language models with the ability to reason over long-range relations. Under the few-shot setting, ReasonBERT in PRIMEQA substantially outperforms RoBERTa (Liu et al., 2019)-based QA systems. PRIMEQA gives any researcher or developer the capability to easily integrate this component in their custom search application e.g. a DPR retriever and a ReasonBERT reader.

Table Readers: Beihang University and Microsoft Research Asia contributed Tapex (Liu et al., 2021) as the first generative Table reader in PRIMEQA. Tapex proposes a novel table pre-training strategy based on a neural SQL executor and achieves SOTA on Wiki-SQL (Zhong et al., 2017a) and Wiki-TableQuestions (Pasupat and Liang, 2015a). They utilize the Transformers (Wolf et al., 2020) sequence-to-sequence trainer for seamless integration into PRIMEQA. LTI CMU’s NeuLab contributed OmniTab (Jiang et al., 2022b), which employs an efficient pre-training strategy leveraging both real and synthetic data. This integration happened organically as OmniTab builds on top of Tapex in PRIMEQA. Currently, their model yields the best few-shot performance on Wiki-TableQuestions, making it also an appropriate candidate system under domain shift.

Custom Search for Earth Science: NASA re-

searchers created a custom search application for scientific abstracts and papers related to Earth Science which received global attention⁹. First, using the top level scripts in PRIMEQA, they trained an OpenQA system on over 100k abstracts by training a ColBERT retriever and an extractive reader. Then, they were able to quickly deploy the search application using the [create-primeqa-app](#) and make it available publicly¹⁰.

6 Conclusion

PRIMEQA is an open-source Question Answering library designed by researchers and developers to easily facilitate reproducibility and reusability of existing and future work in QA. This is an important contribution to the community, as it provides researchers and end users with easy access to state-of-the-art algorithms in the rapidly progressing field of QA. PRIMEQA also provides off-the-shelf models that developers can directly deploy for their custom QA applications. PRIMEQA is built on top of the largest open-source NLP libraries and tools, can incorporate LLMs through external APIs, and provides simple Python scripts as entry points for easy reuse of its core components. This straightforward access and high reusability has already garnered significant traction in the community, enabling PRIMEQA to grow organically as an important resource for rapid progress in QA.

⁹<https://www.nextgov.com/emerging-tech/2023/02/ibm-nasa-will-use-ai-improve-climate-change-research/382437/>

¹⁰<http://primeqa.nasa-impact.net/qa>

Ethics and Broader Impact

6.1 Broader Impact

PRIMEQA is developed as a one-stop and open-source QA repository with an aim to democratize QA research by facilitating easy replication and extension of state of the art methods in multilingual question answering and developments. QA is moving fast with the launch of state-of-the-art (SOTA) retrievers, readers and multi-modal QA models. However, there are two key hurdles which slow the adoption of the SOTA models by the community, which are (1) hard to reproduce for researchers and (2) involves a learning curve for developers to use in custom applications. PRIMEQA solves both the problems by providing multiple access points for different user groups for their easy adoption. PRIMEQA is licensed under Apache 2.0 and thus open to use in both academia and industry. Therefore, PRIMEQA can have an impact on the whole NLP community or more broadly any user working on NLP applications.

6.2 Ethical Considerations

The models available in PRIMEQA might inherit bias based on available training data in public domain. Such bias, if any, is in general present in the models contributed to PRIMEQA and not specific to PrimeQA. Therefore, the usage of PRIMEQA should be approached with the same caution as with any NLP model.

PRIMEQA supports easy access for researchers and developers to use state-of-the-art models and even customize them on their own data. However, PRIMEQA does not control the type of data the model will be exposed to in a custom environment. The general assumption is that these models will be used for rightful purposes in good faith.

Acknowledgments

We thank Lysandre Debut, Thomas Wolf and the Hugging Face team for their close collaboration with the PRIMEQA team to facilitate building PRIMEQA on top of the Hugging Face libraries and datasets. We also thank Chris Potts, Omar Khattab, Jon Saad-Falcon and Keshav Santhanam from Stanford University, Xiang Deng and Huan Sun from Ohio State University, Zhengbao Jiang from Carnegie Mellon University, Qian Liu from Beihang University, and Maximilian Schmidt from University of Stuttgart for contributing implementations of their state-of-the-art systems to PRIMEQA.

References

- Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019a. [Synthetic QA corpora generation with roundtrip consistency](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.
- Chris Alberti, Kenton Lee, and Michael Collins. 2019b. [A bert baseline for the natural questions](#). *arXiv preprint arXiv:1901.08634*.
- Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. [XOR QA: Cross-lingual open-retrieval question answering](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 547–564, Online. Association for Computational Linguistics.
- 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](#). *CoRR*, abs/2005.14165.
- Rishav Chakravarti, Anthony Ferritto, Bhavani Iyer, Lin Pan, Radu Florian, Salim Roukos, and Avirup Sil. 2020. [Towards building a robust industry-scale question answering system](#). In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 90–101.
- Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Jaydeep Sen, Mustafa Canim, Soumen Chakrabarti, Alfio Gliozzo, and Karthik Sankaranarayanan. 2021. [Topic transferable table question answering](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4159–4172, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. [Reading Wikipedia to answer open-domain questions](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.
- Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. [Hybridqa: A dataset of multi-hop question answering over tabular and textual data](#). *Findings of EMNLP 2020*.

- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. [Scaling instruction-finetuned language models](#).
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020a. [TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages](#). *Transactions of the Association for Computational Linguistics*, 8:454–470.
- Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020b. [Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages](#). *Transactions of the Association for Computational Linguistics*, 8:454–470.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Deepset. 2021. [Haystack](#).
- Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. [ReasonBERT: Pre-trained to reason with distant supervision](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, 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, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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.
- Revanth Gangi Reddy, Bhavani Iyer, Md Arafat Sultan, Rong Zhang, Avirup Sil, Vittorio Castelli, Radu Florian, and Salim Roukos. 2021. [Synthetic target domain supervision for open retrieval qa](#). In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1793–1797.
- Revanth Gangi Reddy, Vikas Yadav, Md Arafat Sultan, Martin Franz, Vittorio Castelli, Heng Ji, and Avirup Sil. 2022. [Towards robust neural retrieval with source domain synthetic pre-finetuning](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1065–1070, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. [Allennlp: A deep semantic natural language processing platform](#). *arXiv preprint arXiv:1803.07640*.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020a. [Tapas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020b. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.
- IBM. 2020. [Watson assistant](#).
- Mohit Iyyer, Scott Wen-tau Yih, and Ming-Wei Chang. 2017. [Search-based neural structured learning for sequential question answering](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
- Gautier Izacard and Edouard Grave. 2020. [Leveraging passage retrieval with generative models for open domain question answering](#). In *Conference of the European Chapter of the Association for Computational Linguistics*.

- Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022a. Omnitab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942.
- Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022b. [OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. [Billion-scale similarity search with gpus](#). *CoRR*, abs/1702.08734.
- 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.
- Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. [GooAQ: Open question answering with diverse answer types](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 421–433, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Vishwajeet Kumar, Saneem Chemmengath, Yash Gupta, Jaydeep Sen, Samarth Bharadwaj, and Soumen Chakrabarti. 2021. [Multi-instance training for question answering across table and linked text](#).
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019a. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- 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. 2019b. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.
- 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. 2019c. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.
- Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. [Latent retrieval for weakly supervised open domain question answering](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.
- Yulong Li, Martin Franz, Md Arafat Sultan, Bhavani Iyer, Young-Suk Lee, and Avirup Sil. 2022. [Learning cross-lingual IR from an English retriever](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4428–4436, Seattle, United States. Association for Computational Linguistics.
- Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).
- Shayne Longpre, Yi Lu, and Joachim Daiber. 2021. Mlqa: A linguistically diverse benchmark for multilingual open domain question answering. *Transactions of the Association for Computational Linguistics*, 9:1389–1406.
- Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.
- Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209.
- Scott McCarley, Mihaela Bornea, Sara Rosenthal, Anthony Ferritto, Md Arafat Sultan, Avirup Sil, and Radu Florian. 2023. Gaama 2.0: An integrated system that answers boolean and extractive question. In *AAAI 2023*.
- NVIDIA. 2022. [Tao toolkit](#).

- 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. *arXiv preprint arXiv:1904.01038*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#).
- Panupong Pasupat and Percy Liang. 2015a. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Panupong Pasupat and Percy Liang. 2015b. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. [KILT: a benchmark for knowledge intensive language tasks](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. 21(1).
- 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.
- Marcin-Elvis Guillem Andrew Robert, Ross and Thomas. 2022. [Papers with code](#). In *Meta AI Research*.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Foundations and Trends in Information Retrieval*, 3:333–389.
- Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. [PLAID: An efficient engine for late interaction retrieval](#). In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management, CIKM ’22*, page 1747–1756, New York, NY, USA. Association for Computing Machinery.
- Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. [COLBERTv2: Effective and efficient retrieval via lightweight late interaction](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.
- Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.
- Siamak Shakeri, Noah Constant, Mihir Kale, and Linting Xue. 2021. [Towards zero-shot multilingual synthetic question and answer generation for cross-lingual reading comprehension](#). In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 35–45, Aberdeen, Scotland, UK. Association for Computational Linguistics.
- Md Arafat Sultan, Shubham Chandel, Ramón Fernández Astudillo, and Vittorio Castelli. 2020. [On the importance of diversity in question generation for QA](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.
- Md Arafat Sultan, Avirup Sil, and Radu Florian. 2022. Not to Overfit or Underfit the Source Domains? An Empirical Study of Domain Generalization in Question Answering. In *EMNLP*.
- 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.

Xiang Yue, Ziyu Yao, and Huan Sun. 2022. Synthetic question value estimation for domain adaptation of question answering. In *ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017a. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017b. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Appendix

A.1 PrimeQA Applications

Figure 4 shows a screen-shot of three **PrimeQA** applications. Tables 2 and 3 provide lists of supported datasets and some important PRIMEQA links.

Datasets
OpenNQ
XOR-TyDi (Asai et al., 2021)
SQuAD (Rajpurkar et al., 2016)
TyDiQA (Clark et al., 2020b)
NQ (Kwiatkowski et al., 2019c)
ELI5
SQA (Iyyer et al., 2017)
WTQ (Pasupat and Liang, 2015b)
DocVQA (Mathew et al., 2021)
WikiSQL (Zhong et al., 2017b)

Table 2: A list of some of the supported datasets in PrimeQA

Retriever	Simple Python script
Reader	Inference APIs
Unstructured QG	Inference APIs
Pipeline	Inference APIs

Table 3: Links to PrimeQA

PrimeQA

Retrieval

Question
What are the sources of black carbon in the atmosphere?

5 documents from articles

Title: Document 0
Text: Black carbon, also known as soot, emitted from combustion of fuels and biomass burning absorbs solar radiation in the atmosphere and is one of the major causes of global warming, after carbon dioxide emissions. When black carbon is deposited on snow and ice, the soot-covered snow or ice absorbs more sunlight, leading to surface warming. Due to the large amount of snow and ice in the Arctic—which has warmed twice as fast as the global average over the past century—the region is likely to be especially sensitive to black carbon.

Confidence: 100%
 Score: 27.21875
 Was this document useful?

Title: Document 1
Text: Black carbon, or soot, is the second most important anthropogenic driver of global climate change, taking a backseat only to carbon dioxide. Whether from wood in a cookstove, coal in a power plant, or trees charred by a wildfire, black carbon is produced by the incomplete combustion of organic matter. Once it gets into the environment, black carbon lowers the albedo when it settles on land, increasing warming and enhancing snow and ice melt. In the atmosphere, black carbon both helps and inhibits the formation of clouds.

Confidence: 93%
 Score: 27.140625
 Was this document useful?

PrimeQA

Reading

Demographically, Warsaw was the most diverse city in Poland, with significant numbers of foreign-born residents.[121] In addition to the Polish majority, there was a large and thriving Jewish minority. According to the Imperial Census of 1897, out of the total population of 638,000, Jews constituted 219,000 (equivalent to 34%). [122] Prior to the Second World War, Warsaw hosted the world's second largest Jewish population after New York – approximately 30 percent of the city's total population in the late 1930s.[51] In 1933, 833,500 out of 1,178,914 people declared Polish as their mother tongue.[123] There was also a notable German community.[124] The ethnic composition of contemporary Warsaw is incomparable to the diversity that existed for nearly 300 years.[51] Most of the modern-day population growth is based on internal migration and urbanisation.

Question
How many of Warsaw's inhabitants spoke Polish in 1933?

Answer
833,500

Confidence: 100%
 Was this answer useful?

PrimeQA

Question Answering

Question
What is the name of the Voldemort's snake ?

Found 10 answers matching your question.

Answer
Nagini

Confidence: 83%
 Source: Book7
 Document: Paragraph 1148
 Was this answer useful?

Evidence

Voldemort dropped his gaze once more to the wand in his fingers. It troubled him ... and those things that troubled Lord Voldemort needed to be rearranged. ... "Go and fetch Snape." "Snape, m-my Lord?" "Snape. Now. I need him. There is a — service — I require from him. Go." Frightened, stumbling a little through the gloom, Lucius left the room. Voldemort continued to stand there, twirling the wand between his fingers, staring at it. "It is the only way, **Nagini**," he whispered, and he looked around, and there was the great thick snake, now suspended in midair, twisting gracefully within the enchanted, protected space he had made for her, a starry, transparent sphere somewhere between glittering cage and tank. With a gasp, Harry pulled back and opened his eyes; at the same moment his ears were assaulted with the screeches and cries, the smashes and bangs of battle. "He's in the Shrieking Shack. The snake's with him, it's got some sort of magical protection around it. He's just sent Lucius Malfoy to find Snape."

Figure 4: PrimeQA Applications

Lingxi: A Diversity-aware Chinese Modern Poetry Generation System

Xinran Zhang^{1, 2, 3}, Maosong Sun^{1, 2, 3, 4*}, Jiafeng Liu^{1, 2, 3}, Xiaobing Li^{1, 2, 3}

¹Department of AI Music and Music Information Technology, Central Conservatory of Music

²Key Laboratory of Music and Brain Science, Central Conservatory of Music, Ministry of Education, China

³Laboratory of AI Music, Central Conservatory of Music,

Laboratory of Philosophy and Social Sciences, Ministry of Education, China

⁴Department of Computer Science and Technology, Tsinghua University

zhangxr.wspn@gmail.com, sms@tsinghua.edu.cn

Abstract

Chinese modern poetry generation has been a challenging task. One issue is the Chinese word segmentation (CWS) which is critical to comprehend the Chinese language but was not always considered in common tokenization methods. Another is the decoding (sampling) method which may induce repetition and boredom and severely lower the diversity of the generated poetry. To address these issues, we present *Lingxi*, a diversity-aware Chinese modern poetry generation system. For the CWS issue, we propose a novel framework that incorporates CWS in the tokenization process. The proposed method can achieve a high vocabulary coverage rate with a reasonable vocabulary size. For the decoding method and the diversity issue, we propose a novel sampling algorithm that flattens the high likelihood part of the predicted distribution of the language model to emphasize the comparatively low-likelihood words and increase the diversity of generated poetry. Empirical results show that even when the top 60% of cumulative probability mass of the predicted distribution is flattened, our method achieves comparable or even better performance than baseline sampling methods. Our system is available at <http://lingxi.website>[†].

1 Introduction

Chinese modern poetry generation has been a challenging natural language processing (NLP) task. One issue is the *Chinese word segmentation* (CWS) problem. Unlike English words that are naturally delimited by white spaces, Chinese words do not have explicit word delimiters. Since different CWS strategies may completely change the semantics of Chinese words, CWS is always crucial for humans to comprehend the Chinese language and is regarded as a critical Chinese NLP task. Despite its

importance, CWS is always ignored in benchmark tokenization methods such as byte pair encoding (BPE, Gage, 1994; Sennrich et al., 2016) or unigram language model (Unigram LM, Kudo, 2018), as well as in recent researches on Chinese poetry or lyric generation (e.g., systems by Lee et al., 2019, Zhang et al., 2020, and Zhang et al., 2022). A critical issue is that the rendered vocabulary from CWS tends to exhibit an extremely long “tail”, which demands additional techniques to process before being incorporated into the language model.

Another challenge is the high diversity requirement for poetry generation. The task is unique compared with common neural generation tasks, which try to predict the correct answers from the training corpus and focus on the fluency of the generated texts. Concretely, a piece of easily understood and highly fluent poetry with high-likelihood words might not always be considered poetic, while semantically ambiguous poetry with low-likelihood words might be diversified and creative. We consider the diversity issue from the view of the decoding (sampling) algorithm of the language model. The widely acknowledged golden methods include the nucleus sampling (Holtzman et al., 2020), top-*k* sampling (Fan et al., 2018; Holtzman et al., 2018), and temperature sampling, which have been proved to be able to control the quality and diversity of generated texts. However, in our system, these golden methods might occasionally generate boring and repetitive samples with low diversity, severely hurting the quality of the generated poetry. Thus it inspires us to explore novel decoding algorithms to address the diversity issue for poetry generation.

To address these challenges, we present *Lingxi*, a diversity-aware Chinese modern poetry generation system with the following features. For the CWS issue, we propose a novel framework that incorporates CWS in the tokenization process. The proposed method not only leverages the human knowledge from the CWS model but also com-

*Corresponding author.

[†]Video demonstration is available at <https://youtu.be/ofNTZFCM4DQ>.

binates the advantage of frequency-based tokenization methods, which can achieve a high coverage rate of the vocabulary while maintaining a reasonable vocabulary size. For the high-diversity issue, we propose a novel decoding (sampling) method, namely the *nucleus sampling with flattened head* (NS-FH) algorithm. It flattens the high-likelihood part of the predicted distribution to suppress the high-likelihood words and emphasize the less likely words to improve the diversity of the generated poetry. Surprisingly, we find that even if the top 60% of cumulative probability mass of the vocabulary’s distribution is flattened, the model achieves comparable or even better performance than baseline methods.

2 The CWS Framework for Chinese Modern Poetry Corpus

2.1 The “Long-tail” Issue of CWS

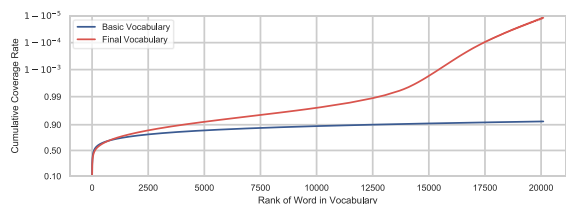


Figure 1: Comparison of the cumulative coverage rate of the vocabulary. The original vocabulary produced by CWS (colored in blue) has a total size of more than 3 million, and the top 20,000 words in the vocabulary has a coverage rate of only 90%. With the proposed framework, the vocabulary coverage rate can be increased to nearly 100% (colored in red) while maintaining a suitable vocabulary size (less than 20,000).

Following the pre-training/fine-tuning paradigm, we collect about 3,500 published books of Chinese novels as the pre-training corpus, aiming at Chinese literary language modeling. Then we collect about 220,000 passages of Chinese modern poetry and lyrics as the fine-tuning corpus. To build the vocabulary, most researchers directly apply frequency-based tokenization methods such as byte pair encoding (BPE, Gage, 1994; Sennrich et al., 2016) or unigram language model (Unigram LM, Kudo, 2018) without considering the Chinese word segmentation (CWS) issue, which is a critical Chinese NLP task and can be modeled by supervised learning (Liu et al., 2014; Yan et al., 2020; Qiu et al., 2020; Duan and Zhao, 2020). Despite these advances, it is still difficult to directly deploy CWS into the tokenization process due to

the “long-tail” issue. Figure 1 illustrates the cumulative coverage rate of the vocabulary (sorted by the word frequency) of our corpus produced by the CWS software THULAC (Sun et al., 2016). It generates a large vocabulary (containing about 3.6 million words) with a very long low-frequency “tail”, which grows slowly to the coverage rate of the vocabulary and takes a nonnegligible portion of coverage. In our case, the top 20,000 words take about 90% coverage on the corpus, and the remaining 3.6 million minus 20,000 words take the remaining 10% coverage, which satisfies Zipf’s law (Zipf, 1949). Truncating the “tail” would result in a low coverage rate and too many “unknown” tokens.

2.2 Proposed CWS Framework

To handle the “long-tail” issue, we propose the following heuristic algorithm, which segments the “tail” into “subwords” using words from the top portion of the vocabulary.

Step 1: Use the CWS tool THULAC to process the corpus and acquire the vocabulary, sort it by the word frequency, and choose a top portion of the vocabulary as the *basic vocabulary*.

Step 2: For the out-of-vocabulary (OOV) words, use the *basic vocabulary* to segment them into *subwords*. For subwords outside the basic vocabulary, add them to the basic vocabulary. If an OOV word has different segmentation strategies, determine by choosing the largest likelihood product of subwords.

Step 3: Sort the *expanded* basic vocabulary, and choose a top portion as the *final vocabulary*.

Let V denote the vocabulary of the corpus produced by the CWS model. For each word $w \in V$, its frequency and likelihood are denoted by $n(w)$ and $p(w)$, which satisfy $p(w) = n(w) / \sum_{w \in V} n(w)$. Sort V by $p(w)$, then choose the top portion of the vocabulary with cumulative coverage rate being P_1 (referred to as “top P_1 ”) to construct the basic vocabulary V_{BASIC} . The rest of words in V (“tail”) are denoted by \bar{V}_{BASIC} . To process \bar{V}_{BASIC} , segment each word in \bar{V}_{BASIC} using words in V_{BASIC} . During this process, a word might have different segmentation strategies. To solve the segmentation ambiguity, let $S(w) = \{w_1, w_2, \dots\}$, $w_i \in V_{BASIC}$ denote a segmentation of word $w \in \bar{V}_{BASIC}$ with an ordered sequence of token w_i . We choose the segmentation strategy with the largest likelihood product,

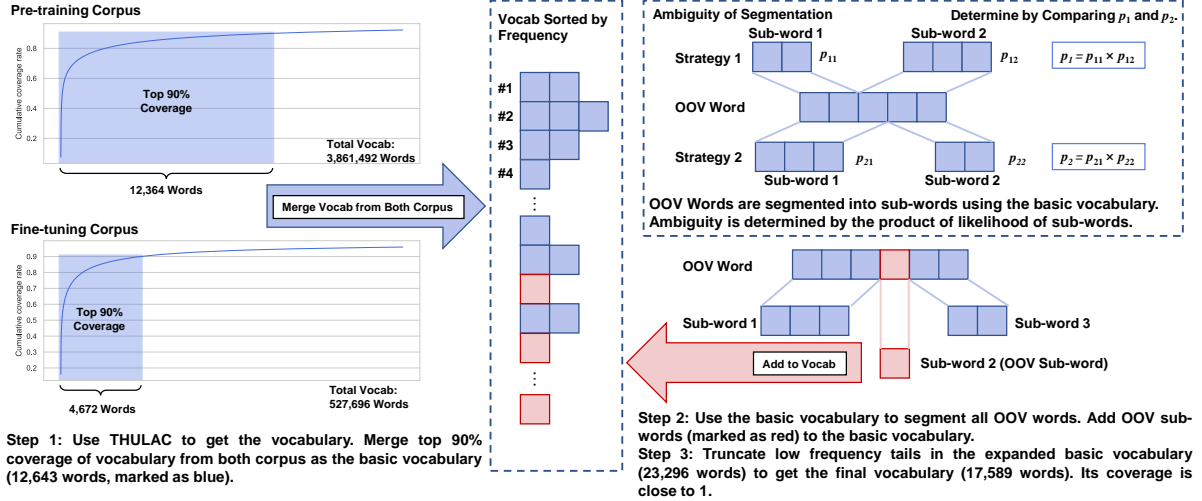


Figure 2: Illustration of the proposed CWS framework for Chinese modern poetry.

i.e., $\operatorname{argmax}_{S(w)} \prod_{w \in S(w)} p(w)$. If any word outside V_{BASIC} is found in the segmentation, add it to V_{BASIC} . After all words in \bar{V}_{BASIC} are processed, V_{BASIC} will be expanded to cover all the corpus. Sort the expanded V_{BASIC} by its updated word likelihood and choose the top P_2 of the vocabulary as the final vocabulary V_{FINAL} . The process is described in Algorithm 1.

Algorithm 1 The proposed CWS Framework

Input: Training Corpus

Output: V_{FINAL}

- 1: Acquire V by THULAC, sort V by $p(w)$, choose top P_1 of V as V_{BASIC} , and its complement as \bar{V}_{BASIC} .
- 2: **for** each $w \in \bar{V}_{BASIC}$ **do**
- 3: Find all possible segmentations $\{S(w)\}$
- 4: $S^*(w) \leftarrow \operatorname{argmax}_{S(w)} \prod_{w \in S(w)} p(w)$
- 5: **for** each $w^* \in S^*(w)$ **do**
- 6: Add w^* to V_{BASIC} **if** $w^* \notin V_{BASIC}$
- 7: Update $p(w)$. Sort V_{BASIC} by $p(w)$, choose top P_2 of V_{BASIC} as V_{FINAL}
- 8: **return** V_{FINAL}

After all words in \bar{V}_{BASIC} are processed, all sub-words outside V_{BASIC} will be added to the vocabulary. In this way, V_{BASIC} will be expanded to cover all information in the corpus, and all OOV words can be segmented into in-vocabulary sub-words. All information will be kept during this process with no “unknown” tokens. And in the last step, only sub-words with a very low word frequency that are rarely used will be filtered out and replaced with “unknown” tokens. In this way,

the final vocabulary will achieve a high coverage rate with a suitable size. The entire process is shown in Figure 2. It combines the CWS model with frequency-based tokenization methods and can generate a vocabulary with suitable size and high coverage rate. In our case, the coverage rate of the final vocabulary is close to 1.0 (larger than $1 - 10^{-4}$, see Figure 1), with a vocabulary size being 17,589.

Since we have two different training corpora, we seek to leverage the word frequency feature from both corpora. For **Step 1**, we choose the top 90% of the vocabulary from a) both corpora and b) the fine-tuning corpus only, then merge them as the basic vocabulary. This emphasizes the fine-tuning corpus to benefit its generation task. The size of basic vocabulary merged in **Step 1** is 12,643. After **Step 2**, the vocabulary size is expanded to 23,296. In **Step 3**, we drop words with a frequency lower than a) 100 on both corpora and b) 10 on the fine-tuning corpus and acquire the final vocabulary with 17,589 words. By observation, the dropped words in **Step 3** are all extremely rare single-length Chinese words.

3 The Diversity-aware Sampling

We train an auto-regressive Transformer language model on the two collected corpus for generation (see the appendices for the model details). As is widely acknowledged, the decoding module plays a crucial role in neural text generation. Stochastic sampling methods such as nucleus sampling (Holtzman et al., 2020), top- k sampling (Fan et al., 2018; Holtzman et al., 2018) and temperature sam-

pling can generate texts with higher diversity than traditional decoding methods such as beam search. Recent advances such as MIROSTAT (Basu et al., 2021) and typical sampling (Meister et al., 2023) use adaptive filtering on top of these methods. However, in our model, we observe that traditional algorithms occasionally generate boring and repetitive poetry with low diversity, thus severely hurting the quality of generated poetry.

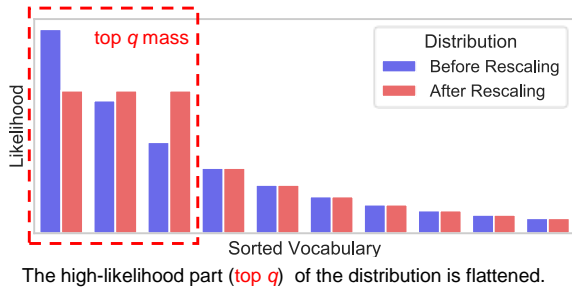


Figure 3: Illustration of the proposed NS-FH algorithm, which flattens and rescales the high-likelihood part of the predicted distribution to emphasize less likely words and increase the diversity of generated poetry.

Recent research has revealed that the low-diversity issue can stem from the high-likelihood part of the predicted distribution (Holtzman et al., 2020; Basu et al., 2021), which is not fully leveraged by traditional sampling methods. We consider the intuition for poetry composing that fluent poetry with too many high-likelihood words can be boring (i.e., the boredom trap, Basu et al., 2021); in contrast, semantically ambiguous poetry with surprising and low-likelihood words can be creative and poetic (e.g., poems by James Joyce or Marcel Proust). Inspired by this, we propose a novel sampling algorithm to emphasize the less likely words on the predicted distribution to increase the diversity. We leverage the notion of nucleus sampling (NS) by defining an additional filtering parameter denoted by q to identify the high-likelihood part (“head”) of the vocabulary, denoted by V^{head} . Then we propose to *flatten and evenly redistribute* the probability mass for V^{head} , to emphasize the “comparatively low likelihood” words in the “head”. For the low-likelihood part of the distribution (“tail”), we adopt nucleus sampling with parameter p ($p \geq q$) to truncate the “tail” like the tradition. Stochastic sampling is conducted on the flattened and rescaled distribution for all sampling steps. The above method is referred to as *nucleus sampling with flattened head* (NS-FH) algorithm. The diversity gain of the algorithm is

controlled by q , which determines the boundary of V^{head} . The algorithm is illustrated in Figure 3.

The proposed algorithm has a close relationship with nucleus sampling. Their truncation mechanisms on the predicted distribution are identically based on the threshold $\{p, q\}$ of the cumulative probability density function. And our method features in using an additional threshold q to determine another smaller portion of the predicted distribution and flattening their probability density function. And since q is also a threshold for the cumulative probability density function, the flattening manipulation will not affect low-entropy distributions that only contain an “unquestionably correct” word which takes most of the probability mass, thus not hurting the fluency of the generated poetry. We later show in the empirical results that the proposed method can closely follow the behavior of nucleus sampling when using a small value of q while exhibiting higher diversity when setting a large value of q , achieving diversity-aware sampling.

4 Demonstration and Evaluation

4.1 System Interface

The system interface and poetry example are shown in Figure 4. We provide a plain mode and an advanced mode. For the plain mode, input the poetry prompt, choose the diversity parameter with the slider, and hit the “submit” button to get a generated poetry. By default, the system uses the proposed NS-FH algorithm as the decoding method. In the advanced mode, we implemented the three golden sampling methods (nucleus sampling, top- k sampling, and temperature sampling) as well as two novel sampling methods (MIROSTAT by Basu et al., 2021 and typical sampling by Meister et al., 2023) for users to choose and compare.

4.2 Impact of Sampling Algorithms

It is noteworthy that the performance of the sampling algorithm is highly dependent on the sampling hyperparameter. To illustrate and compare their impact on the generated poetry, we take advantage of the quality-diversity trade-off feature (Nadeem et al., 2020; Zhang et al., 2021; Basu et al., 2021). By tuning the hyperparameter for one sampling algorithm, the quality (fluency) metric and the diversity metric of the generated poetry form a trade-off curve in which an increase in the diversity metric will decrease the fluency metric.

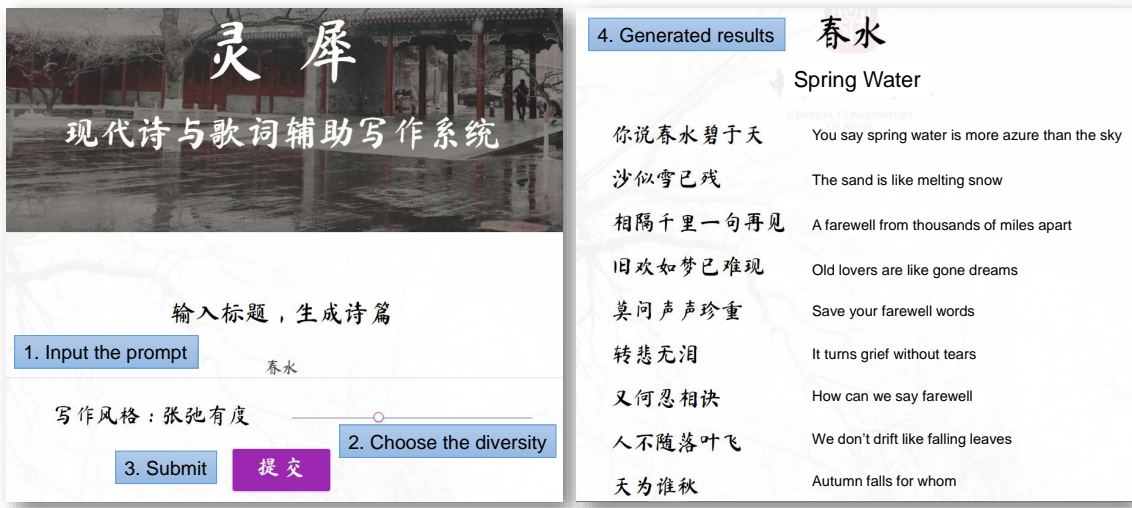


Figure 4: The system interface and generated poetry sample of Lingxi.

Such a feature demands a holistic picture of the trade-off curve by traversing the hyperparameter space and evaluating automatic metrics on the generated poetry to reveal the nature of the algorithm. As a result, we leverage the trade-off feature by aligning corresponding metrics on a 2D plane throughout our evaluation.

Concretely, we use every poetry title from the validation set as the input prompt for generation, which results in 15,218 generations per sampling algorithm per hyperparameter. We choose the fluency metric as the perplexity (PPL) of generated poetry (lower score indicates higher fluency but more boredom). We then choose the following three diversity metrics: the Zipf coefficient (Zipf, 1949; Newman, 2005) which reflects the sloping tendency of the word frequency distribution (lower score indicates a flatter distribution and higher diversity); the entropy of n -gram distribution (Zhang et al., 2018) which reflects the diversity of n -grams (higher score indicates less repeating n -grams and higher diversity); the self-BLEU score (Zhu et al., 2018; Holtzman et al., 2020) which reflects the overlapping tendency among different generated poetry passages (lower score indicates less overlapping and higher diversity). We calculate the human-level metrics from the validation set as the reference point.

Results for the fluency-diversity trade-off curves are shown from Figure 5 to Figure 7. We show regions around the human (reference) metric point for a clear view (see Figure 10 to Figure 14 in the appendix for full curves). Results show that the human-level metric of Chinese modern poetry

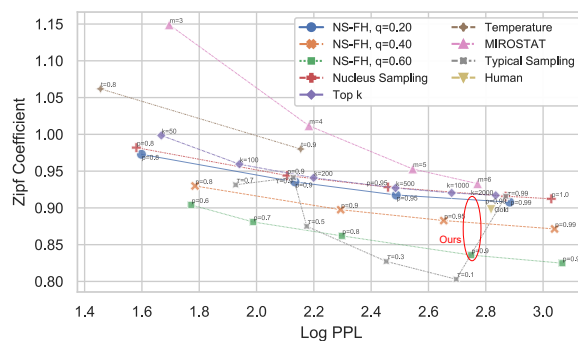


Figure 5: The trade-off curve for Zipf coefficient (diversity \downarrow) against perplexity. Although all methods loosely converge to the human (reference) point, traditional methods must greatly relax their sampling parameters ($p = 0.99$, $k = 2000$, $t > 0.9$, $m > 6$, $\tau = 0.99$), i.e., almost degrade to pure sampling to achieve human-level metrics. With a small diversity parameter, our method (NS-FH, $q=0.20$) has a similar trajectory to nucleus sampling; with a large diversity parameter, our method (NS-FH, $q=0.60$) exhibits higher diversity (lower curve) than other methods in most cases (only except for typical sampling with $\tau = 0.10$), achieving diversity-aware sampling.

suggests a **high diversity requirement** (low Zipf coefficient, high entropy of n -gram, and low self-BLEU score). To achieve human-level diversity, traditional methods must greatly relax their sampling parameter and almost degrade to pure sampling (sampling on the original predicted distribution with no filtering) for maximum diversity. By contrast, our method controls the trade-off curve by tuning the diversity parameter q without fully relaxing the filtering. When using a small value of $q = 0.20$, our method closely follows the trajectory

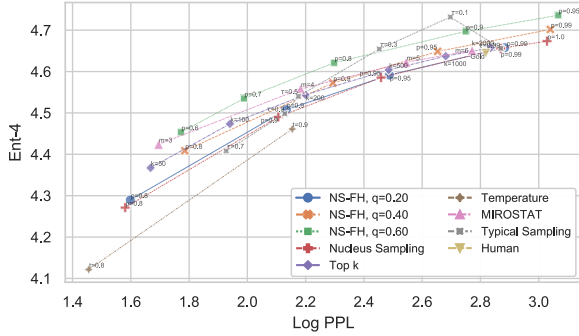


Figure 6: The trade-off curve for the entropy of 4-gram distribution (Ent-4, diversity \uparrow) against perplexity. All methods exhibit close trajectories with each other and converge to the **human** (reference) point. Our method (NS-FH, $q=0.60$) can also achieve the diversity upper bound achieved by typical sampling with $\tau = 0.1$, meanwhile its trade-off curve still stays close to the reference point.

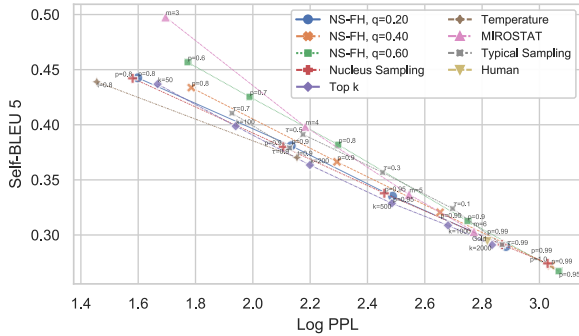


Figure 7: The trade-off curve for self-BLEU 5 (diversity \downarrow) against perplexity. While all methods exhibit close trajectories near the **human** (reference) point, our method (NS-FH, $q=0.60$) achieves the highest diversity boundary (lowest metric in the lower right region).

of nucleus sampling and converges to the reference point; when using a large value of $q = 0.60$, our method achieves a comparable or higher diversity metric to traditional methods without deviating from the reference point. These results indicate that our method can achieve diversity-aware sampling with good metric performance.

Note a very interesting behavior of the NS-FH algorithm that by setting $q = 0.60$ (colored in green), i.e., the top 60% of cumulative probability mass of the predicted distribution is completely flattened and ignored, the trade-off curve does not drift away from the human-level metric point (see the *performance degradation* of sampling algorithms illustrated in Figure 3 by Nadeem et al., 2020), while even achieving higher diversity metrics and boundaries in most cases. This suggests that in our task with a high requirement for diversity, the probability of

high-likelihood words on a “flat” distribution with high entropy can be *ignored*. Completely flattening the distribution and ignoring the predicted probability of high-likelihood words can counter-intuitively achieve comparable or better results with higher diversity. By inference, such a method might be suitable for other artistic generation tasks like music or drawings that require high diversity.

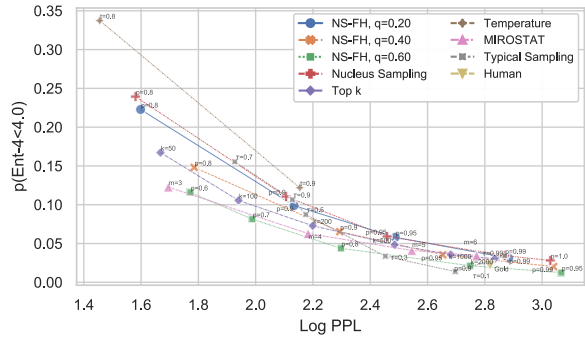


Figure 8: The trade-off curve for $p(\text{Ent-4} < 4.0)$ (\downarrow) against perplexity. Similar to the results of previous diversity metrics, our method achieves a lower metric boundary than traditional methods (except for typical sampling with $\tau \leq 0.3$) and still converges to the **human** (reference) point.

Also, note the stochastic nature of the sampling methods, which yields a stochastic trajectory of tokens with variational quality. Under the same sampling parameter and conditions, the generated poetry might occasionally be repetitive and wordy. So we focus on the distribution of the entropy of n -gram metric to investigate the repetition tendency. We calculate $p(\text{Ent-4} < \eta)$, which reflects the chance that the generated poetry has the entropy of n -grams lower than the threshold η . Empirically, we set $\eta = 4$ to get a meaningful observation. Results are shown in Figure 8. They show that naive sampling parameters for traditional methods easily result in more repetition, e.g., nucleus sampling with $p = 0.80$ has $p(\text{Ent-4} < 4) \approx 0.24$. Similar to previous results, traditional methods have to greatly relax the filtering parameter or degrade to pure sampling to approach human-level repetition tendency. By contrast, our method with $q = 0.60$ achieves a lower curve (less repetition chance) than most methods and also stay close to the reference point. This indicates that our method with a high diversity parameter can grant less repetition as well as maintain a close relationship of repetition tendency to human behavior without fully relaxing the filtering.

4.3 Rhyming Feature

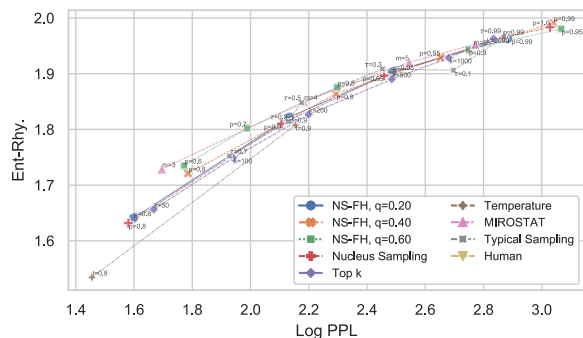


Figure 9: The trade-off curve for the entropy of rhyming word distribution against the fluency metric. Results show that our method has on-par rhyming performance with all traditional methods.

We also consider an additional metric to reflect the rhyming feature of the generated poetry. Similar to the entropy metric, we calculate the entropy of rhyming word distribution by $\mathbb{E}\{-\log p_{rhyme}(x)\}$, where $p_{rhyme}(x)$ is the rhyming word frequency distribution (higher score indicates higher diversity but less rhymed). Similarly, we plot the fluency-rhyming trade-off curves in Figure 9. The results show that our method achieves on-par rhyming performance with all baseline methods. It indicates that the flattening manipulation of our method does not hurt the rhyming feature of the generated poetry, despite that our method flattens the top portion of the cumulative probability mass of the predicted distribution.

Since rhyming is an important feature for composing Chinese modern poetry, we provide an additional function to control the rhyme of the generated poetry. In the advanced mode of LingXi, we adopt a re-ranking and replacing mechanism for rhyming. During the generation process, once a [NEWLINE] symbol was generated, we try to replace its previous token with a rhyming one. We create hypotheses by combining each token from the vocabulary that fits the requirement of the chosen rhyme with a [NEWLINE] token, recalculate the average perplexity of the combined tokens (hypothesis), and choose the hypothesis with the lowest average perplexity to replace the generated one. One issue is that there might be cases in which all hypotheses have high perplexity (low likelihood). So we set an additional threshold that the hypothesis must have an average perplexity lower than the threshold to trigger the replacement. If none of the hypotheses meet the requirement of the thresh-

old, the generation remains unrevised. The above method can achieve a trade-off between rhyming and fluency by tuning the perplexity threshold. Empirically, we set the threshold to 50 to achieve a good performance. Although the method cannot guarantee that all generated lines of poetry meet the rhyming requirement, it features in its flexibility to plug into all decoding methods implemented in our system without modifying the language model or the algorithms.

5 Conclusion

We present Lingxi, a diversity-aware Chinese modern poetry generation system. To address the CWS issue, we propose a novel framework that combines CWS with the frequency-based method, which can create a vocabulary with a suitable size and high coverage. To increase the diversity of generated poetry, we propose nucleus sampling with flattened head (NS-FH) algorithm which achieves controllable diversity with on-par or better performance compared to traditional sampling methods. The proposed sampling algorithm provides a new approach to increase the diversity of neural text generation via the decoding module, which might be beneficial for artistic generation cases that have high requirements for the diversity or novelty.

Acknowledgments

This work has been funded by: (i) Special Program of National Natural Science Foundation of China (Grant No. T2341003), (ii) Advanced Discipline Construction Project of Beijing Universities, (iii) Major Program of National Social Science Fund of China (Grant No. 21ZD19), (iv) Nation Culture and Tourism Technological Innovation Engineering Project (Research and Application of 3D Music), (v) National Philosophy and Social Science Project (Grant No. 22VJXG012).

References

- Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. 2021. *MIROSTAT: A neural text decoding algorithm that directly controls perplexity*. In *International Conference on Learning Representations*.
- Sufeng Duan and Hai Zhao. 2020. *Attention is all you need for Chinese word segmentation*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3862–3872, Online. Association for Computational Linguistics.

- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. **Hierarchical neural story generation**. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.
- Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. **The curious case of neural text de-generation**. In *International Conference on Learning Representations*.
- Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. **Learning to write with cooperative discriminators**. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2014. **Adam: A method for stochastic optimization**.
- 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.
- Hsin-Pei Lee, Jhih-Sheng Fang, and Wei-Yun Ma. 2019. **iComposer: An automatic songwriting system for Chinese popular music**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 84–88, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. **Domain adaptation for CRF-based Chinese word segmentation using free annotations**. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874, Doha, Qatar. Association for Computational Linguistics.
- Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. **Locally Typical Sampling**. *Transactions of the Association for Computational Linguistics*, 11:102–121.
- Moin Nadeem, Tianxing He, Kyunghyun Cho, and James Glass. 2020. **A systematic characterization of sampling algorithms for open-ended language generation**. 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 334–346, Suzhou, China. Association for Computational Linguistics.
- Mark EJ Newman. 2005. Power laws, pareto distributions and zipf’s law. *Contemporary physics*, 46(5):323–351.
- Xipeng Qiu, Hengzhi Pei, Hang Yan, and Xuanjing Huang. 2020. **A concise model for multi-criteria Chinese word segmentation with transformer encoder**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2887–2897, Online. 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 (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. **Thulac: An efficient lexical analyzer for chinese**. <http://thulac.thunlp.org/>.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. **Huggingface’s transformers: State-of-the-art natural language processing**. *ArXiv*, abs/1910.03771.
- Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. **A graph-based model for joint Chinese word segmentation and dependency parsing**. *Transactions of the Association for Computational Linguistics*, 8:78–92.
- Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. **Trading off diversity and quality in natural language generation**. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.
- Le Zhang, Rongsheng Zhang, Xiaoxi Mao, and Yongzhu Chang. 2022. **QiuNiu: A Chinese lyrics generation system with passage-level input**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 76–82, Dublin, Ireland. Association for Computational Linguistics.
- Rongsheng Zhang, Xiaoxi Mao, Le Li, Lin Jiang, Lin Chen, Zhiwei Hu, Yadong Xi, Changjie Fan, and Minlie Huang. 2020. **Youling: an AI-assisted lyrics creation system**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 85–91, Online. Association for Computational Linguistics.
- Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. **Generating informative and diverse conversational**

responses via adversarial information maximization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. [Txygen: A benchmarking platform for text generation models](#). In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '18*, pages 1097–1100, New York, NY, USA. Association for Computing Machinery.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

A Model Configuration, Training Details and Supplementary Results

Detailed configurations and parameters of our model are listed in Table 1. For special tokens, the [TITLE] token is added directly after the title of each poetry passage in the fine-tuning corpus to capture the title feature. The [START-OF-PASSAGE] token is added before the starting token of the first poetry line. We create a replica for each poetry passage excluding the title and [TITLE] token, and mix them with the original corpus as data augmentation. The [NEWLINE] token is added at the end of each poetry line in replace of the newline character, and the [END-OF-PASSAGE] token is added at the end of each poetry passage. English words and letters are assigned [UNK-EW] tokens. Other unknown words and sub-words are assigned [UNK] tokens. For poetry passages longer than the maximum context length, we create training samples using a sliding window with stride being half of the maximum context length. We split train/validation/test sets using the common ratio of 85%/7%/8% (token ratio for the pre-training corpus, passage ratio for the fine-tuning corpus).

The model is an auto-regressive Transformer decoder, using cross entropy as the training loss. The training process is developed using the Huggingface library by [Wolf et al. \(2019\)](#). It achieves monotonic convergence of perplexity (PPL) on the validation set of the pre-training corpus at the end of the pre-training steps. We choose the best fine-tuning epoch of the model with the lowest PPL on the validation set of the fine-tuning corpus as the final model. For the generated poetry, their full metric trade-off curves for all sampling hyperparameters and sampling algorithms are shown from [Figure 10](#) to [Figure 14](#).

Parameters	Value
number of Transformer layers	24
number of Transformer attention heads	16
embedding size	1,024
vocabulary size	17,589
maximum context length	128
number of network parameters	330 million
pre-training epochs	20
fine-tuning epochs	10
batch size per GPU	32
number of training GPUs	8 NVIDIA [®] GeForce [®] RTX 2080 Ti
pre-training learning rate	2×10^{-4}
fine-tuning learning rate	$\{1 \times 10^{-5}, 2 \times 10^{-5}\}$
learning rate decay	linear decay
warm-up steps	1% of total steps
optimizer	Adam optimizer (Kingma and Ba, 2014)
weight decay	0.01
PPL on validation set after pre-training	17.58
best fine-tuning epoch	epoch 4, learning rate being 2×10^{-5}
best PPL on fine-tuning validation set	16.75
full sampling hyperparameter space for the trade-off curves from Figure 10 to Figure 14.	$p \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 1\}$ $q \in \{0.2, 0.4, 0.6\}, p \geq q$ $k \in \{10, 20, 50, 100, 200, 500, 1000, 2000\}$ $t \in \{0.6, 0.7, 0.8, 0.9, 1.1\}$ $m \in \{2, 3, 4, 5, 6\}$ $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$

Table 1: Model configuration, training details and sampling parameters

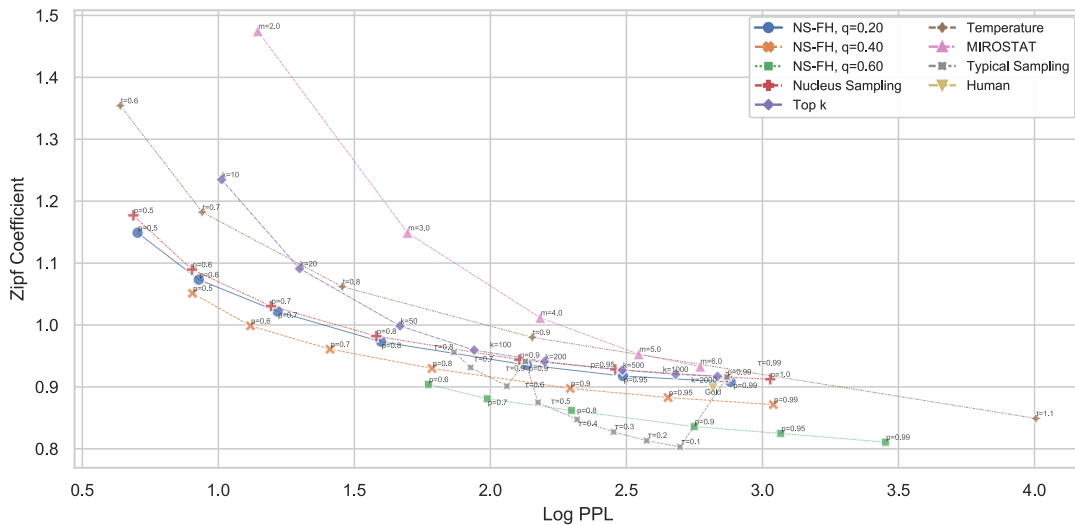


Figure 10: Full trade-off curve for Zipf coefficient against perplexity.

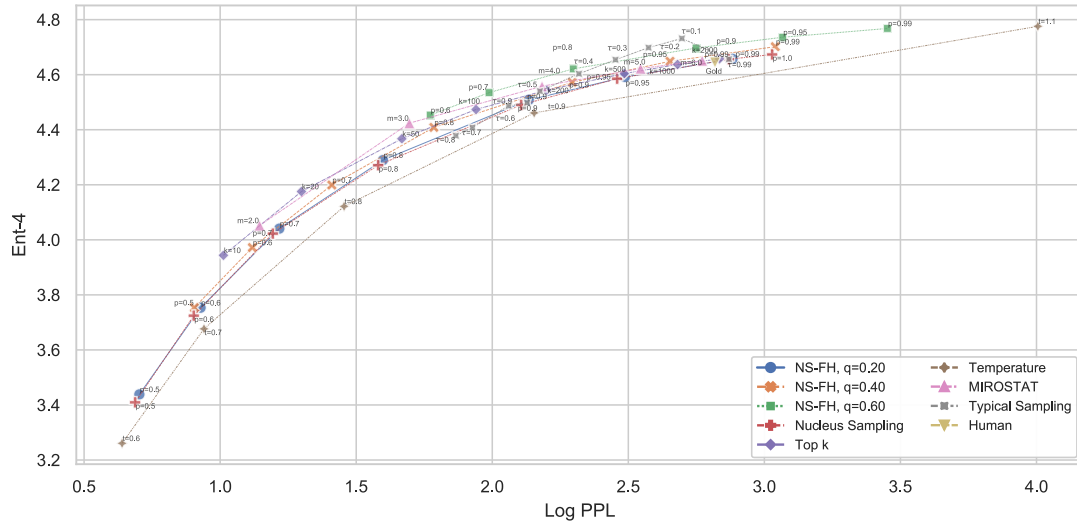


Figure 11: Full trade-off curve for the entropy of 4-gram distribution (Ent-4) against perplexity.

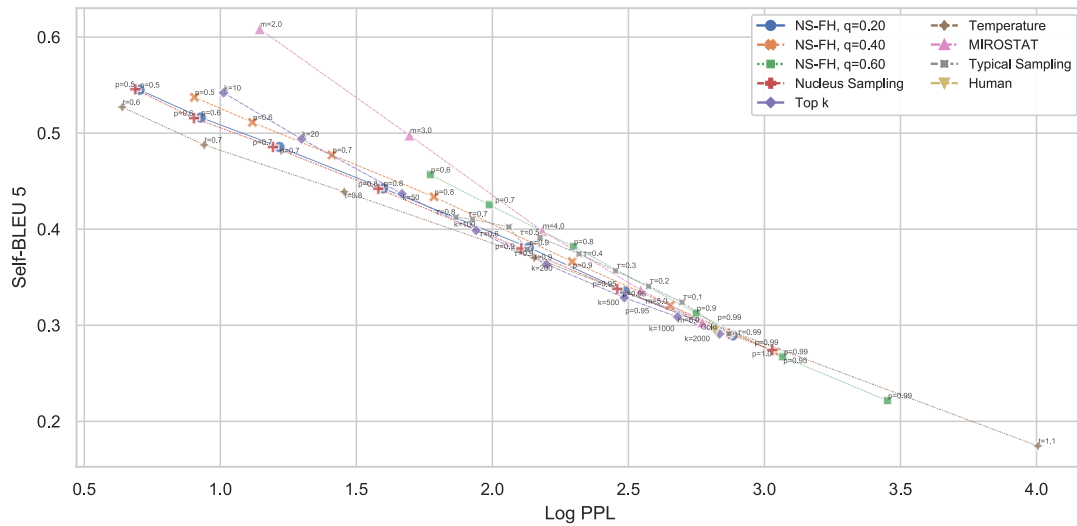


Figure 12: Full trade-off curve for self-BLEU 5 against perplexity.

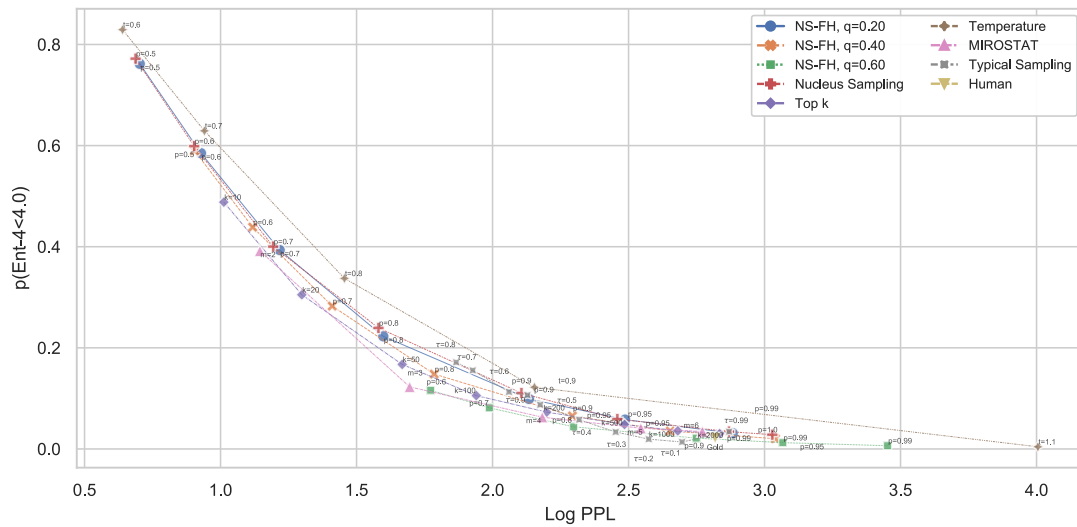


Figure 13: Full trade-off curve for $p(\text{Ent-4} < 4.0)$ against perplexity.

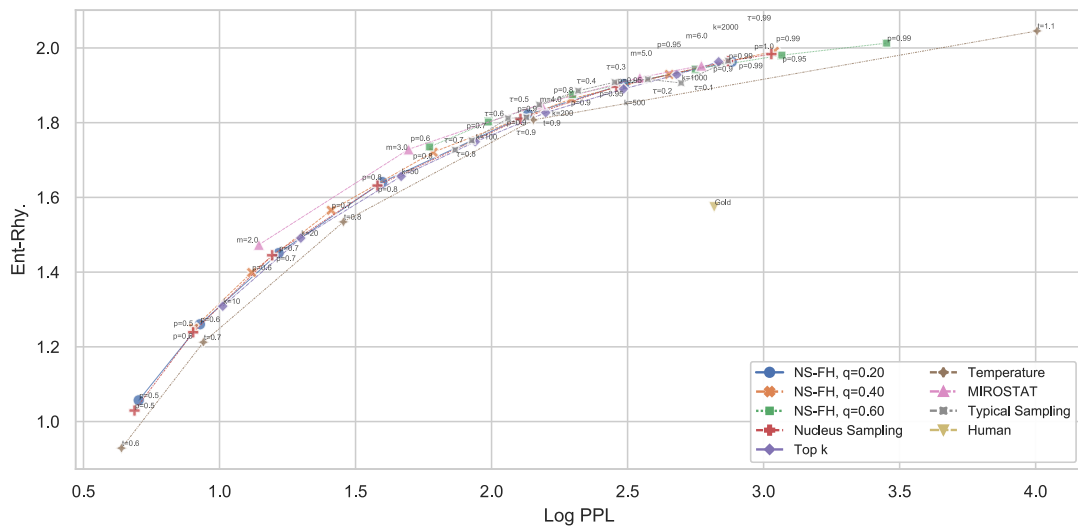


Figure 14: Full trade-off curve for the entropy of rhyme distribution against the fluency metric.

Autodive: An Integrated Onsite Scientific Literature Annotation Tool

Yi Du^{1,2,3*}, Ludi Wang¹, Mengyi Huang^{1,2}, Dongze Song¹, Wenjuan Cui^{1,2},
Yuanchun Zhou^{1,2,3*}

¹ Computer Network Information Center, Chinese Academy of Sciences

² University of Chinese Academy of Sciences

³ University of Science and Technology of China

{duyi, wld, myhuang, songdongze, wenjuancui, zyc}@cnic.cn

Abstract

Scientific literature is always available in Adobe's Portable Document Format (PDF), which is friendly for scientists to read. Compared with raw text, annotating directly on PDF documents can greatly improve the labeling efficiency of scientists whose annotation costs are very high. In this paper, we present Autodive, an integrated onsite scientific literature annotation tool for natural scientists and Natural Language Processing (NLP) researchers. This tool provides six core functions of annotation that support the whole lifecycle of corpus generation including i)annotation project management, ii)resource management, iii)ontology management, iv>manual annotation, v)onsite auto annotation, and vi)annotation task statistic. Two experiments are carried out to verify efficiency of the presented tool. A live demo of Autodive is available at <http://autodive.sciwiki.cn>, and a video demo <http://autodive.sciwiki.cn/introVideo/introduce-v1.0.mp4>. The source code is available at <https://github.com/Autodive>.

1 Introduction

Influential applications such as AlphaFold2 (Jumper et al., 2021), and mobile robotic chemist (Burger et al., 2020) rely on high-quality databases and domain knowledge, some of which are constructed from scientific literature. The traditional method of building such databases still needs the manual annotation of numerous professionals. With the development of scientific research and the enhancement of interdisciplinary integration, the number of scientific articles has increased explosively, which also requires the annotators who are assigned to build the domain-specific database to have a suitable background and are familiar with annotation of literature data. Some research attempts to assist the manual annotation process with the

intelligent natural language model, such as Named Entity Recognition (NER) and Relation Identification (RI). The intelligent method can automatically extract knowledge from articles, and form high-quality databases after expert proofreading (Cruse et al., 2022; Yan et al., 2022). However, the performance of the general model in specific fields is not satisfactory, and the construction of a specific intelligent model needs high-quality databases. Thus, An easy-to-use annotation tool with a graphical user interface that allows the labeling of text efficiently and consistently is crucial and necessary.

Annotation tools play a crucial role during the database-making process in the field of biology (López-Fernández et al., 2013), material (Corvi et al., 2021), and chemistry (Swain and Cole, 2016). Although the majority of released annotation tools mainly focus on the annotation of multimedia such as image and video, there are still text annotation tools such as AnnIE (Friedrich et al., 2021), TS-ANNO (Stodden and Kallmeyer, 2022) and Doccano (Nakayama et al., 2018). However, most of the tools need to convert the input file from PDF format to plain text, which may cause additional labor costs for resource preparation. Moreover, the annotation of specific data required professionals with knowledge of different fields. Compared with raw text, annotating directly on PDF documents conforms to the reading habits of the professionals with the original images, tables and layout information and can greatly improve the labeling efficiency of them whose annotation costs are very high. We refer to this need as **Onsite Annotation**, which includes the abilities to display and direct annotate on the original PDF documents.

To meet the needs of professionals on direct annotation, the new tool should support direct scientific literature annotation in PDF format. That means this tool should display the original literature in PDF format directly, so users can read the complete PDF content in the annotation inter-

*Correspond Author

face, where they annotate the entity and relation of scientific literature through scheduled annotation operation. In this way, this tool retains the layout information of the original PDF literature to fit the reading habits of annotators. Different from traditional text annotation projects, scientific annotation projects require annotation tools to process long literature, annotate complex entities and export appropriate annotation results to a formed database.

Besides onsite annotation, integration is also a valuable factor during the design of the tool. First, the existing domain-specific ontology should be easily integrated which can reduce the time costs of ontology design. Second, the tool should be well integrated with existing and new Named-entity Recognition (NER) models flexibly. Third, integration with existing file systems or literature collections, such as Pubmed Central and self-organized document folders, should also be valued. By integrating with an existing data source, a researcher can deploy this tool locally without leakage of unpublished or copyrighted documents.

Hence, we propose an open-sourced annotation tool Autodive, with its contributions summarized below:

(1) Onsite Annotation. Autodive supports onsite annotation of PDF documents for professionals. They can annotate directly on PDF documents, and get instant visual feedback.

(2) Integration. Autodive can integrated external modules that may assist the whole annotation process, including corpus management, ontology construction, manual and intelligent annotation.

(3) Domain Verified. The effectiveness has been verified by two tasks, including catalytic material annotation and scientific dataset annotation.

2 Architecture

The overall architecture of Autodive is shown in Fig.1 with three layers which are Data Source, Server Layer, and Frontend. The core component of Autodive is the **Server Layer**. *Data Source Adapter* can integrate specific **Data Sources** such as Pubmed Central or File System with Autodive. Three core server-side engines play important roles. The first is the *Regular Expression Parsing Engine*. It can help Autodive extract entities by predefined regular expressions. The second is the *Ontology Management Engine*. It can load and parse ontology files with OWL format and save the designed

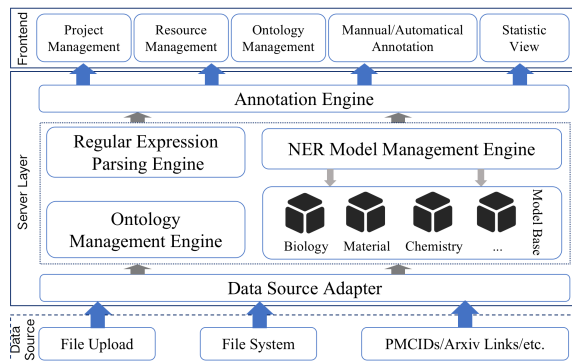


Figure 1: Architecture of Autodive.

ontology to OWL or self-defined JSON files. The third engine is the *NER Model Management Engine*. It is a model base that is extendable and friendly to newly trained NER models. The annotation engine is the bridge between the server-side and the frontend. This engine provides locating and feedback of entities. **Frontend** is implemented mainly using a progressive JavaScript framework Vue.js.

As shown in Fig.2, the complete literature in PDF format is shown on the right part of the annotation page of Autodive, where annotators can read the literature and annotate the entity. Users click the mouse to select words on the displayed PDF document, and click the right mouse to select the type of entities or relationships. When connecting two annotated entities, users can annotate relation of them. Shortcut-key annotation is allowed for the increase of efficiency of the annotation project. All identified or annotated entities and relations are listed in the entity-label-list and relation-label-list with predefined ontology.

3 Modules

Autodive integrates management, annotation, and optimization through six modules and is specially built for scientific literature annotation projects. Users use the **Project Management** module to develop and manage personalized annotation projects. The **Resource Management** module allows users to manage their own literature resources pool. The **Ontology Management** module pre-defines the knowledge ontology, which is defined and standardized by the annotation project administrator. The basic components for annotating are the **Manual Annotation** module and the **Auto Annotation** module. Trained automatic annotation models are saved in the background in order to improve the productivity of manual annotation work. Finally,

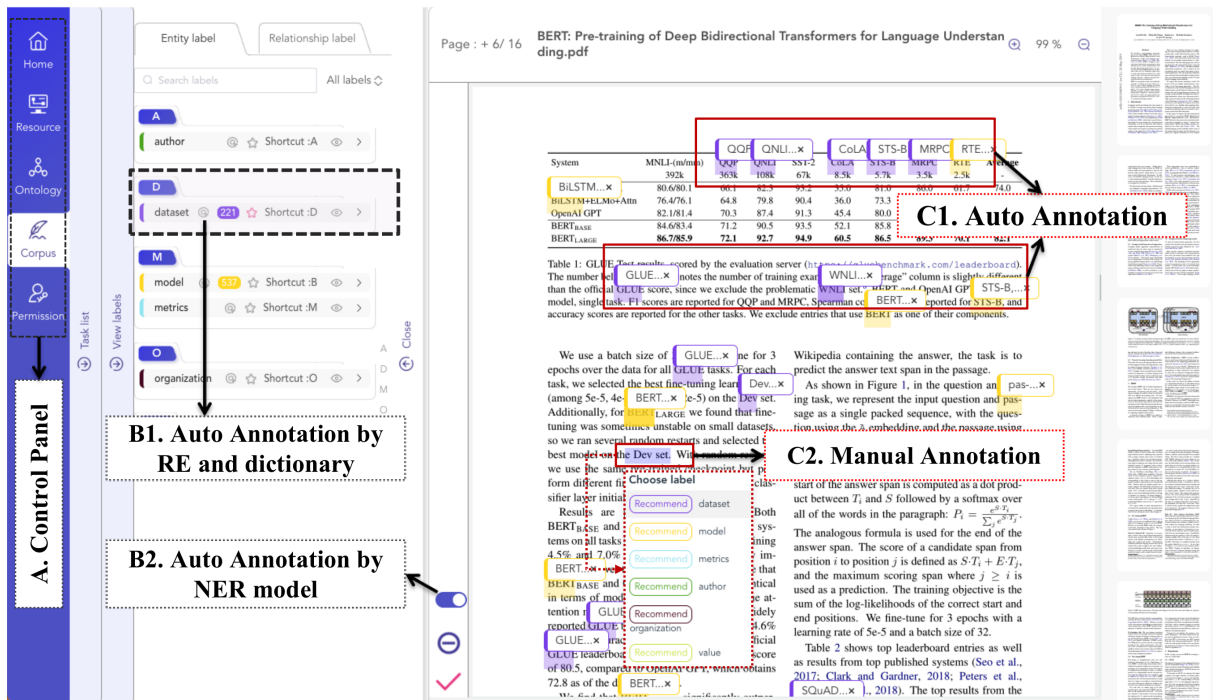


Figure 2: Overview of Autodive annotation interface. The left part of the interface is A. Control Panel that integrates all functions of Autodive. B1 and B2 demonstrate usage of three auto annotation functions, while C1 and C2 show the results of both auto annotation(C1) and manual annotation(C2).

we created a **Statistic View** module for Autodive to represent the overall progress of the annotation project.

3.1 Project Management

The annotation lifecycle starts with the creation of one annotation project. Autodive designs project management module that includes project creation and annotator administration. As shown in Fig.3(a), users submit necessary information such as project name while creating an annotation project. Domain information is also encouraged to fill out so that the auto-annotation model can accelerate the annotation.

There are three different kinds of roles. The creator of the project, also plays as the administrator, has the ability to invite annotators. If the invitation email is approved, invitees will be added straight to the relevant annotation project as administrators or annotators. The administrator also has the rights to assign annotation tasks to project members.

3.2 Resource Management

For a literature annotation project, the initial and critical step is to control which documents to be annotated and where to find them. The quantity and diversity of literature affect the quality of annotation data, and therefore the accuracy of intelligent

models. Autodive allows users to upload and manage their own literature resources, and form a list of documents relating different annotation projects by associating the literature resources with them. Besides uploading PDF documents directly, this tool also provides standard API (Application Programming Interface) that can import scientists' own literature resources.

After initialing the list of annotation resources, the administrator can assign the annotation resources to other annotators, and the annotators can complete the follow-up task in the form of crowdsourcing, which is as shown in Fig.3(b).

3.3 Ontology management

Ontology means what kinds of entities and relations to be annotated in the annotation project, which is defined and controlled by the project administrator to fit task requirements. Well defined ontology can enhance the efficiency of annotation. The process of ontology design is shown in Fig.3(c). The first basis component of ontology management is load and design of one ontology. Considering the complexity of the link between entity-labels, the same relation-label category may distribute to numerous entity-labels pairs, and multiple relation-labels may distribute to the same entity-labels pair. The

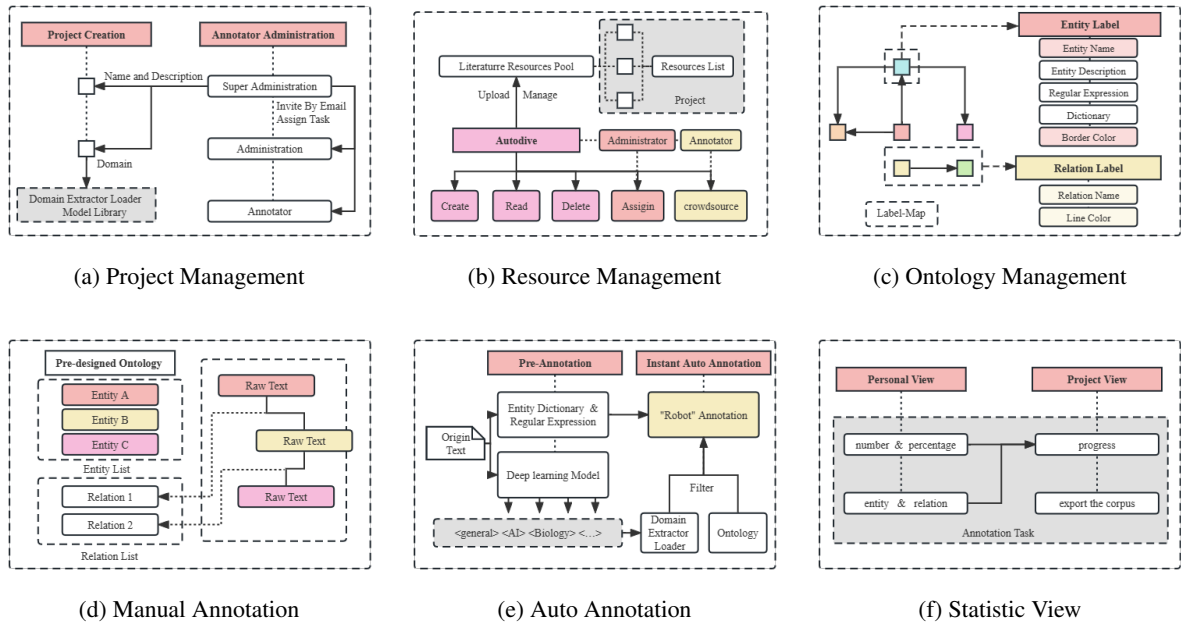


Figure 3: Operation process of six core modules in Autodive.

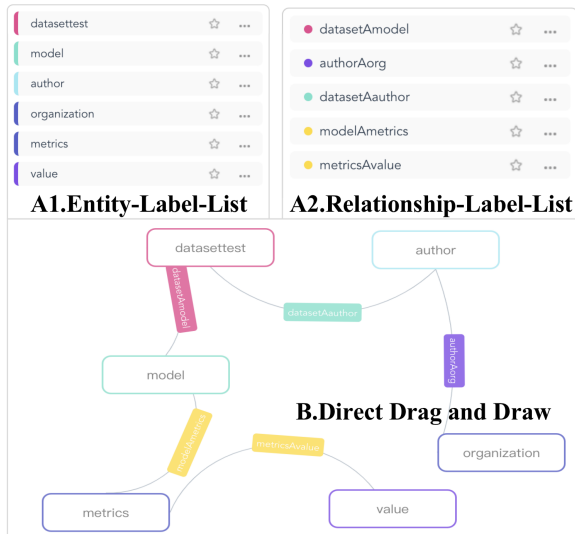


Figure 4: Two different display modes of ontology management.

label-map will display everything mentioned above. Autodive also allow users download the designed ontology for subsequent research or application.

During the design process of ontology, Autodive allow users upload the corresponding dictionary and fill up the required regular expression. It is advantageous to simplify some annotation projects when the entity-label has a clear thesaurus to-be-annotated terms or words that have a unified structure. Two display modes of ontology is shown in Fig.4.

3.4 Manual Annotation

The manual annotation of an entity in Autodive can select raw text instead of drawing a bounding box in PDF document, this mode is not like PAWLS(Neumann et al., 2021). After selecting, the user decides what kind of entity the text is. This way of entity annotation is more precise, especially in annotating text that has line wrap in document. The manual annotation also provide a controlled annotation of relationship. Annotator select two annotated entity, then Autodive will recommend possible relationship that pre-designed in ontology. This recommendation step can also increase the efficiency of annotation.

3.5 Auto Annotation

Autodive provides pre-annotation by three ways, they are regular expression parsing, dictionary mapping, and external NER models. It is an evident advantage for dictionary and regular expression annotation since they basically annotate "the proper terms". However, when there are unavoidable out-of-vocabulary (OOV) words in the to-be-annotated literature, intelligent model annotation using external NER models is a more effective choice.

It is evident that in various scientific research domains, the model of private domain learned the typical information during training, allowing it to accurately recognize the label. Autodive uses the **Domain Extractor Loader** and the **Model Li-**

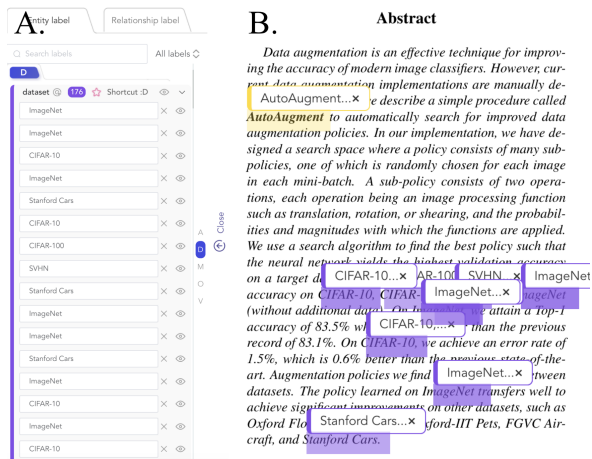


Figure 5: Auto annotation. A shows list of auto extracted labels. B displays instant auto annotation and onsite feedback.

brary to choose specific domain prediction models. When starting an annotation project, users should select the corresponding field of the project. In advance, we saved annotation models in several scientific domains in model library. Furthermore, when the project reaches unexcited fields, users are prompted to select a model from the "generic domain" and fill up the domain details.

One highlight function of the auto annotation is **Instant Auto Annotation**. As shown in Fig.5, the annotation results can be presented right on the complete text with a single click. To support this function, Autodive constructed a robot annotation layer, and the annotation on the source document is displayed in the same way as the manual annotation results. The text parsing tool will first import the text into the auto annotation model, and the model will return the annotation results. The label will then be filtered according to the ontology established by the project, and it will be matched to the precise spot on the document. The auto annotation model adds a robot annotation layer to the literature display layer. The deep learning model will undoubtedly take some time. Autodive chooses to extract literature in advance during background free time, saves the automatic annotation results in the database, and then performs specific matching annotation based on the model domain and ontology chosen by each announcer, reducing the waiting time of specific users.

3.6 Statistic View

To help the administrator and annotator know the status and progress of one annotation task, Auto-

diver provides **Personal View** and **Project View** in this module. In Personal View, the number and percentage of current annotation task are provided so that annotator can evaluate his/her task. Besides, the number of annotated and auto-recognized entity and relationship is also shown in the view. In Project View, functions are provided to help administrator understand current progress of all annotators and their assigned tasks, such as the distribution of annotated entity and relationship, number and percentage of each annotator and his/her task, and so on. In Project View, user can export the corpus for further use.

4 Evaluation & Case Study

4.1 Annotation Tools Evaluation

We compared Autodive with other open sourced annotation tools, including AnnIE(Friedrich et al., 2021), Doccano(Nakayama et al., 2018), WebAnno(Yimam et al., 2013), INCEpTION(Klie et al., 2018), PDFAnno(Shindo et al., 2018) and PAWLS(Neumann et al., 2021), for annotation function comparison.

In order to match the need of scientific literature annotation, we design the evaluation metrics as bellows: The first is **[A].Availability**, which includes *[A1].Activity* and *[A2].Online Service*. The second is **[B].Onsite Support**, which includes *[B1].Onsite PDF Display* and *[B2].Onsite PDF Annotation*. The third is **[C].Function Integration**, which includes *[C1].Integration with File System*, *[C2].Integration with Ontology*, and *[C3].Integration with Pre-annotation Model*. The last is other functions such as **[D].Team Annotation** and **[E].Statistics**. A deeper description of these metrics is given in [Definition of Evaluation Metrics](#).

The comparison results are shown in Tab.1. As shown in Tab.1, Autodive is superior to most active tools in the function of onsite PDF annotation. PDFAnno, PAWLS, and INCEpTION have functions for PDF annotation. However, PDFAnno has not been maintained for over 3 years. Compared with PAWLS, Autodive provides more integration functions with file systems and NER models which also depends on onsite annotation mode. Autodive provides a different integration mode with pre-annotation models and a more intuitive statistics view when compared with the latest version of INCEpTION.

Tools	Availability		Onsite		Integration			Team	Statistics
	[A1]	[A2]	[B1]	[B2]	[C1]	[C2]	[C3]	[D]	[E]
AnnIE	-	-	-	-	-	✓	-	-	-
Doccano	✓	✓	-	-	✓	✓	url	✓	✓
WebAnno	-	-	-	-	✓	✓	-	-	-
PDFAnno	-	-	✓	block	-	-	-	-	-
PAWLS	-	✓	✓	block	-	✓	✓	-	-
INCEpTION(*)	✓	✓	✓	text	✓	✓	url	✓	-
Autodive(ours)	✓(*)	✓	✓	text	✓	✓	✓	✓	✓

Table 1: Function Comparison of Text Annotation tools. Autodive will remain active (A1) and expand to support more formats, such as iamges and tables in documents. Onsite PDF Annotation (B2) have two different modes: annotating by drawing a block(**block**) and annotating by selecting a raw text(**text**). Integration with Pre-annotation Model (C3) have two different modes: directly using an external API(**url**) or integrated with a pre-trained model(✓). (*) the version of INCEpTION we compared is the latest stable version V26.8.

4.2 NER of Cu-based Electrocatalysts

A high-quality corpus of catalysts may assist domain scientists in the discovery of catalysts based on a descriptor-optimization (Tran and Ulissi, 2018; Zhong et al., 2020). In this case, a corpus of Cu-based electrocatalysts for CO₂ reduction has been generated using the presented tool. At the beginning of the process, one senior scientist creates an annotation project and served as the administrator of the project by **Project Management Module**. After creating the project, the scientist finds the literature that needs annotation and assigns the literature to potential annotators using **Resource Management Module**. At the same time, he designs the ontology of Cu-based electrocatalysts with this assist of **Ontology Management Module**. There are 5 postgraduates with experience in experimental catalysis who used the tool to construct the corpus using **Manual Annotation Module** and **Auto Annotation Module**. During the annotation process, the administrator and annotators can view the rate of progress at any time by **Statistic View**. After annotation, the senior scientist exports the corpus and review all the annotated entity and relationship. In this real case, the corpus contains a collection of 6,086 records extracted from 835 publications with nine types of knowledge, including material, regulation method, product, faradaic efficiency, cell setup, electrolyte, synthesis method, current density, and voltage. This annotated corpus can be accessed publicly(Wang, 2023)(Wang et al., 2023).

4.3 Auto Annotation of AI Dataset and Model

In this case, we try to demonstrate the ability of auto annotation. A basic auto annotation project

requires related model prepared and continuous annotation data. In order to analyse the effect of **Auto Annotation Module** quantitatively, we designed an experiment with a poorly correlated public dataset and increased proportion of annotated data to evaluate the correctness of auto annotation, as in a real annotation project. Firstly we trained an annotation model using the SciERC(Luan et al., 2018) dataset that mainly focused on the field of artificial intelligence. After deploying the model to the Autodive backend, we chose a number of abstracts from publications in the field of artificial intelligence on paperswithcode.com to simulate the automatic annotation effect. We designed ontology with "Model" and "Dataset" entities. All data contains 7,420 "Datasets" entities and 42,696 "Model" entities.

Training and updating the NER model during the annotation process helps fit the annotation project and improve the correctness. Tab.2 displays the results of this simulation. The zero-shot shows the correctness without any "annotation" data. With the updates of the auto annotation model through increasing sample size, the increased correctness of auto annotation module shows effectiveness of Auto Annotation Module, which helps annotators train their auto annotation models. As we can see, well integrated auto annotation model might meet the needs of scientific literature annotation, and it may perform better in specific projects.

5 Related Work

Mariana Neves(Neves and Ševa, 2021) gave a comprehensive review of existing document annotation tools. It splits the criteria of document annotation

Tools	Sample Size			
	zero-shot	0.2	0.5	0.8
Autodive	0.32	0.55	0.82	0.90

Table 2: Experiment Result

tools into four categories, which are publication, technical, data, and functional. In this review, it rates WebAnno(Yimam et al., 2013) as the best tool, which also extends to a new human-in-the-loop tool INCEpTION(Klie et al., 2018). It also mentioned that the top two missing functions of current tools are the support of document-level annotation, integration with existing corpus and pre-annotation, especially model-based pre-annotation.

Kinds of new annotation tools are reported in recent years. Many tools focus on specific tasks or functions, such as TeamTat(Islamaj et al., 2020), QuickGraph(Bikaun et al., 2022), DoTAT(Lin et al., 2022) and FAST(Kawamoto et al., 2021). Both task specified and common document annotation tools such as WebAnno(Yimam et al., 2013), Doccano(Nakayama et al., 2018), AnIE(Friedrich et al., 2021) and TS-ANNO(Stodden and Kallmeyer, 2022) need a pre-process that convert document to pure text, which is a time-consuming work. By consulting with domain experts, the converter process also causes confuses reading, especially in the typeset document such as scientific literature.

As for annotation tools for PDF documents, PDFAnno(Shindo et al., 2018) and PAWLS(Neumann et al., 2021) are the two most relevant tools with our present tool. PDFAnno converts PDF documents into pure text without retaining PDF structure information, whose annotation mode is similar to our tool. However, it has not been maintained for over 3 years. PAWLS is a recent tool that supports PDF annotation with labels and structure. It has the advantage of annotation the meta or structural information by drawing a bounding box rather than selecting raw text. Autodive is inspired by PAWLS in the requirement of PDF annotation and surpasses it in integration function and annotation mode.

6 Discussion

We created Autodive, a collaborative scientific literature annotation tool that offers a comprehensive solution for the whole lifecycle of annotation, especially for scientific literature annotation. In

addition, we provide automated annotation for annotators and can integrated with a variety of NER models. We found that Autodive can not only be utilized for scientific literature, but also for any editable PDF file. Also, Autodive is released as an open source project under Apache 2.0 license.

Autodive also has some limitations. First, accuracy of NER models. For text annotation projects, a more accurate NER model can ensure the accuracy of auto annotation. Second, collision in teamwork. Reduce annotation disputes in annotation projects caused by diverse knowledge perspectives between annotators and obtain more accurate annotation data. Third, there are PDF files that cannot be changed. Some obsolete or illegible PDF documents are difficult to process and cannot be used in the current Autodive version.

In our future project, we intend to expend the scope of literature annotation to include graphics and tables in addition to text. Additionally, autodive will support more file types, including the ability to download in JSON/CoNLL format and to upload plain text and pictures.

Acknowledgements

We thank all anonymous reviewers. This work is supported by the National Key Research and Development Plan of China under Grant No. 2022YFF0712200 and 2022YFF0711900, the Natural Science Foundation of China under Grant No. 61836013, Beijing Natural Science Foundation under Grant No. 4212030, Youth Innovation Promotion Association CAS.

References

- Tyler Bikaun, Michael Stewart, and Wei Liu. 2022. [Quickgraph: a rapid annotation tool for knowledge graph extraction from technical text](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.
- Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. 2020. [A mobile robotic chemist](#). *Nature*, 583(7815):237–241.
- Javier Corvi, Carla Fuenteslópez, José Fernández, Josep Gelpi, Maria-Pau Ginebra, Salvador Capella-Guitierrez, and Osnat Hakimi. 2021. [The biomaterials annotator: a system for ontology-based concept annotation of biomaterials text](#). In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 36–48.

- Kevin Cruse, Amalie Trewartha, Sanghoon Lee, Zheren Wang, Haoyan Huo, Tanjin He, Olga Kononova, Anubhav Jain, and Gerbrand Ceder. 2022. **Textmined dataset of gold nanoparticle synthesis procedures, morphologies, and size entities.** *Scientific Data*, 9(1):1–12.
- Niklas Friedrich, Kiril Gashteovski, Mingying Yu, Bhushan Kotnis, Carolin Lawrence, Mathias Niepert, and Goran Glavaš. 2021. **Annie: an annotation platform for constructing complete open information extraction benchmark.** *arXiv preprint arXiv:2109.07464*.
- Rezarta Islamaj, Dongseop Kwon, Sun Kim, and Zhiyong Lu. 2020. **Teamtat: a collaborative text annotation tool.** *Nucleic acids research*, 48(W1):W5–W11.
- John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, et al. 2021. **Highly accurate protein structure prediction with alphafold.** *Nature*, 596(7873):583–589.
- Shunyo Kawamoto, Yu Sawai, Kohei Wakimoto, and Peinan Zhang. 2021. **Fast: fast annotation tool for smart devices.** In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 372–381.
- Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. **The inception platform: Machine-assisted and knowledge-oriented interactive annotation.** In *proceedings of the 27th international conference on computational linguistics: system demonstrations*, pages 5–9.
- Yupian Lin, Tong Ruan, Ming Liang, Tingting Cai, Wen Du, and Yi Wang. 2022. **Dotat: a domain-oriented text annotation tool.** In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–8.
- Hugo López-Fernández, Miguel Reboiro-Jato, Daniel Glez-Peña, Fernando Aparicio, Diego Gachet, Manuel Buenaga, and Florentino Fdez-Riverola. 2013. **Bioannotate: a software platform for annotating biomedical documents with application in medical learning environments.** *Computer methods and programs in biomedicine*, 111(1):139–147.
- Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. **Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction.** In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232.
- Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. **doccano: text annotation tool for human.** Software available from <https://github.com/doccano/doccano>.
- Mark Neumann, Zejiang Shen, and Sam Skjonsberg. 2021. **Pauls: Pdf annotation with labels and structure.** In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, page 258–264.
- Mariana Neves and Jurica Ševa. 2021. **An extensive review of tools for manual annotation of documents.** *Briefings in bioinformatics*, 22(1):146–163.
- Hiroyuki Shindo, Yohei Munesada, and Yuji Matsumoto. 2018. **Pdfanno: a web-based linguistic annotation tool for pdf documents.** In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.
- Regina Stodden and Laura Kallmeyer. 2022. **TS-ANNO: an annotation tool to build, annotate and evaluate text simplification corpora.** In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 145–155, Dublin, Ireland. Association for Computational Linguistics.
- Matthew C Swain and Jacqueline M Cole. 2016. **Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature.** *Journal of chemical information and modeling*, 56(10):1894–1904.
- Kevin Tran and Zachary W Ulissi. 2018. **Active learning across intermetallics to guide discovery of electrocatalysts for co2 reduction and h2 evolution.** *Nature Catalysis*, 1(9):696–703.
- Ludi Wang. 2023. **A corpus of CO2 Electrocatalytic Reduction Process extracted from the scientific literature.**
- Ludi Wang, Yang Gao, Xueqing Chen, Wenjuan Cui, Yuanchun Zhou, Xinying Luo, Shuaishuai Xu, Yi Du, and Bin Wang. 2023. **A corpus of co2 electrocatalytic reduction process extracted from the scientific literature.** *Scientific Data*, 10(1):175.
- Rongen Yan, Xue Jiang, Weiren Wang, Depeng Dang, and Yanjing Su. 2022. **Materials information extraction via automatically generated corpus.** *Scientific Data*, 9(1):1–12.
- Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. **Webanno: a flexible, web-based and visually supported system for distributed annotations.** In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6.
- Miao Zhong, Kevin Tran, Yimeng Min, Chuanhao Wang, Ziyun Wang, Cao-Thang Dinh, Phil De Luna, Zongqian Yu, Armin Sedighian Rasouli, Peter Brodersen, et al. 2020. **Accelerated discovery of co2 electrocatalysts using active machine learning.** *Nature*, 581(7807):178–183.

A Definition of Evaluation Metrics

The definition of evaluation metrics among kinds of annotation tools are shown belows.

[A] Availability. **[A1] Activeness :** The annotation tool is still active and updated steadily. **[A2] Code Availability:** The annotation tool is open sourced.

[B] Onsite Support. **[B1] Onsite PDF Display:** The annotation tool provides a complete display with the structural information of the literature to suit the reading habits of scientific annotators. **[B2] Onsite PDF Annotation:** The annotation tool provides direct annotation on PDF documents, including **Text PDF Annotation** (annotate text directly) and **Block PDF Annotation** (annotate by drawing frames).

[C] Function Integration. **[C1] Integration With File System (Resource Management):** The annotation tools has a file system which allows resource management such as uploading files. **[C2] Integration With Ontology:** The annotation tool enables the definition of ontology such as knowledge graph. **[C3] Integration With Pre-Annotation Model:** The annotation tool offers pre-annotation such as dictionary matching.

[D] Team Annotation: The annotation tool enables team annotation and the management of annotation results for all.

[E] Statistics: The annotation tool provides statistic view of project.

B Page Design

B.1 Project Management

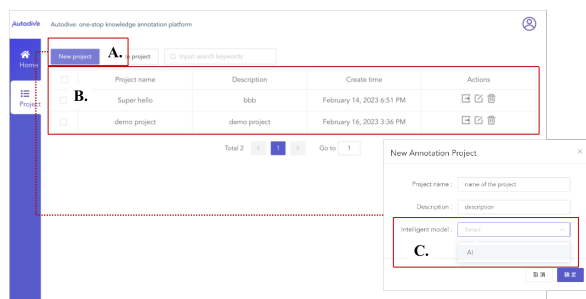


Figure 6: Screenshot of project management. A is a button of creating new annotation project. Once click "New Project" button, user can input project and choose pre-annotation model, just like C. B shows the list of created annotation projects.

B.2 Resource Management

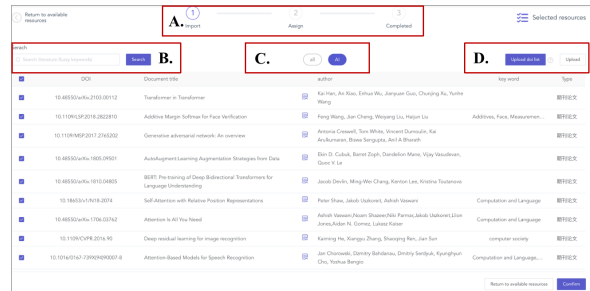


Figure 7: Screenshot of resource management. A shows the three steps of literature search and assign. B, C, D gave different ways to find or import literature files, such as direct search, using file tags or direct upload.

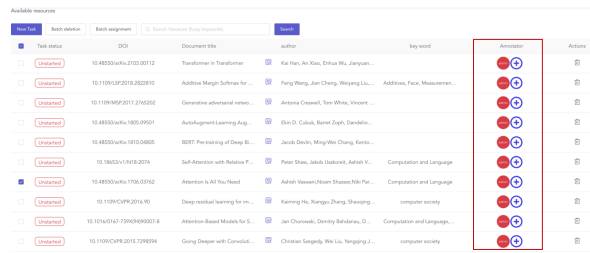


Figure 8: Screenshot of annotation task assignment. Highlighted part shows the function of assignment. Administrator of one project have permission to assign literature to different annotators.

B.3 Ontology Management

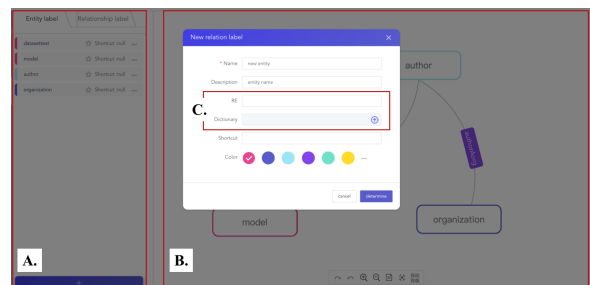


Figure 9: Screenshot of ontology management. A displays the list of all generated entities and relationships. B visualizes the constructed ontology. C is the interactive interface of entity design. In this interface, project owner can define a regular expression or upload a dictionary, so that Autodive can pre-annotate by the regular expression or dictionary.

A Practical Toolkit for Multilingual Question and Answer Generation

Asahi Ushio and Fernando Alva-Manchego and Jose Camacho-Collados
Cardiff NLP, School of Computer Science and Informatics, Cardiff University, UK
{UshioA,AlvaManchegoF,CamachoColladosJ}@cardiff.ac.uk

Abstract

Generating questions along with associated answers from a text has applications in several domains, such as creating reading comprehension tests for students, or improving document search by providing auxiliary questions and answers based on the query. Training models for question and answer generation (QAG) is not straightforward due to the expected structured output (i.e. a list of question and answer pairs), as it requires more than generating a single sentence. This results in a small number of publicly accessible QAG models. In this paper, we introduce AutoQG, an online service for multilingual QAG, along with `lmqg`, an all-in-one Python package for model fine-tuning, generation, and evaluation. We also release QAG models in eight languages fine-tuned on a few variants of pre-trained encoder-decoder language models, which can be used online via AutoQG or locally via `lmqg`. With these resources, practitioners of any level can benefit from a toolkit that includes a web interface for end users, and easy-to-use code for developers who require custom models or fine-grained controls for generation.

1 Introduction

Question and answer generation (QAG) is a text generation task seeking to output a list of question-answer pairs based on a given paragraph or sentence (i.e. the context). It has been used in many NLP applications, including unsupervised question answering modeling (Lewis et al., 2019; Zhang and Bansal, 2019; Puri et al., 2020), fact-checking (Ousidhoum et al., 2022), semantic role labeling (Pyatkin et al., 2021), and as an educational tool (Heilman and Smith, 2010; Lindberg et al., 2013). The most analysed setting in the literature, however, has been question generation (QG) with pre-defined answers, as this simplifies the task and makes the evaluation more straightforward.

Despite its versatility, QAG remains a challenging task due to the difficulty of generating compo-

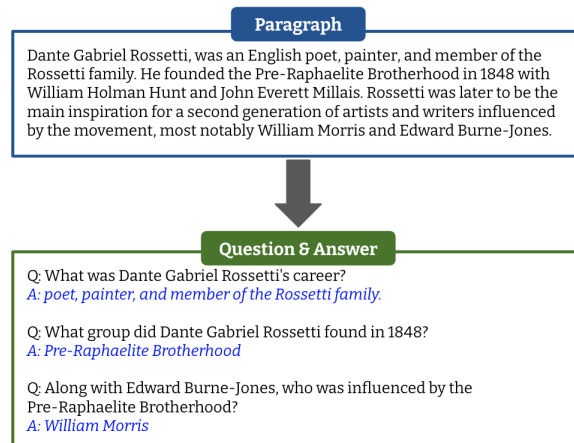


Figure 1: An example of question and answer generation given a paragraph as context.

sitional outputs containing a list of question and answer pairs as shown in Figure 1, with recent works mainly relying on extended pipelines that include several ad-hoc models (Lewis et al., 2021; Bartolo et al., 2021). These works integrate QAG into their in-house software, preventing models to be publicly released, and their complex pipelines make them hard to reproduce and use by practitioners.

In this paper, we introduce an open set of software tools and resources to assist on the development and employment of QAG models for different types of users. We publicly release the following resources:¹

- `lmqg`,² a Python package for QAG model fine-tuning and inference on encoder-decoder language models (LMs), as well as evaluation scripts, and a deployment API hosting QAG models for developers;

¹All the resources except for the datasets are released under an open MIT license, while the datasets follow the license of their original release.

²<https://github.com/asahi417/lm-question-generation>

- 16 models for English, and three diverse models for each of the seven languages integrated into our library, all fine-tuned on QG-Bench (Ushio et al., 2022) and available on the HuggingFace hub (Wolf et al., 2020);³
- *AutoQG* (<https://autoqg.net>), a website where developers and end users can interact with our multilingual QAG models.

2 Resources: Models and Datasets

Our QAG toolkit makes use of pre-existing models and datasets, fully compatible with the HuggingFace hub. This makes our library easily extendable in the future as newer datasets and better models emerge. In this section, we describe the datasets (§ 2.1) and models (§ 2.2) currently available through `lmqg` and *AutoQG*.

2.1 Multilingual Datasets

Our toolkit integrates all QG datasets available in QG-Bench (Ushio et al., 2022). QG-Bench is a multilingual QG benchmark consisting of a suite of unified QG datasets in different languages. In particular, we integrate the following datasets: SQuAD (English), SQuADShifts (Miller et al., 2020) (English), SubjQA (Bjerva et al., 2020) (English), JAQuAD (So et al., 2022) (Japanese), GerQuAD (Möller et al., 2021) (German), SberQuAd (Efimov et al., 2020) (Russian), KorQuAD (Lim et al., 2019) (Korean), FQuAD (d’Hoffschmidt et al., 2020) (French), Spanish SQuAD (Casimiro Pio et al., 2019) (Spanish), and Italian SQuAD (Croce et al., 2018) (Italian). QG-Bench is available through our official `lmqg` HuggingFace project page and GitHub⁴.

2.2 Models

Aiming to make QAG models publicly accessible in several languages, we used `lmqg` to fine-tune LMs using QG-Bench (§ 2.1). First, we defined a pipeline QAG model architecture consisting of two independent models: one for answer extraction (AE) and one for question generation (QG). During training, the AE model learns to find an answer in each sentence of a given paragraph, while the QG model learns to generate a question given an answer from a paragraph. To generate question-answer pairs at generation time, the AE model

first extracts answers from all the sentences in a given paragraph, and then these are used by the QG model to generate a question for each answer. While not directly evaluated in this paper, we also integrated other types of QAG methods such as multitask and end2end QAG (Ushio et al., 2023), all available via the `lmqg` library (§ 3) as well as *AutoQG* (§ 5).

As pre-trained LMs, we integrated T5 (Raffel et al., 2020), Flan-T5 (Chung et al., 2022), and BART (Lewis et al., 2020) for English; and mT5 (Xue et al., 2021) and mBART (Liu et al., 2020) for non-English QAG models. The pre-trained weights were taken from checkpoints available in the HuggingFace Hub as below:

- `t5-{small,base,large}`
- `google/flan-t5-{small,base,large}`
- `facebook/bart-{base,large}`
- `google/mt5-{small,base}`
- `facebook/mbart-large-cc25`

All the fine-tuned QAG models are publicly available in our official HuggingFace Hub. While we initially integrated these models, users can easily fine-tune others using `lmqg`, as we show in § 3.

3 `lmqg`: An All-in-one QAG Toolkit

In this section, we introduce `lmqg` (Language Model for Question Generation), a Python library for fine-tuning LMs on QAG (§ 3.1), generating question-answer pairs (§ 3.2), and evaluating QAG models (§ 3.3). Additionally, with `lmqg`, we build a REST API to host QAG models to generate question and answer interactively (§ 5). `lmqg` is interoperable with the HuggingFace ecosystem, as it can directly make use of the datasets and models already shared on the HuggingFace Hub.

3.1 QAG Model Fine-tuning

Fine-tuning is performed via `GridSearcher`, a class to run encoder-decoder LM fine-tuning with hyper-parameter optimization (see Appendix A for more details). For example, the following code shows how we can fine-tune T5 (Raffel et al., 2020) on SQuAD (Rajpurkar et al., 2016), with the QAG model explained in § 2.2. Since we decomposed QAG into AE and QG, two models need to be fine-tuned independently.

```
from lmqg import GridSearcher
```

³<https://huggingface.co/lmqg>

⁴https://github.com/asahi417/lm-question-generation/blob/master/QG_BENCH.md

```

# instantiate AE trainer
trainer_ae = GridSearcher(
    dataset_path="lmqg/qg_squad",
    input_types="paragraph_sentence",
    output_types="answer",
    model="t5-large")

# train AE model
trainer_ae.train()

# instantiate QG trainer
trainer_qg = GridSearcher(
    dataset_path="lmqg/qg_squad",
    input_types="paragraph_answer",
    output_types="question",
    model="t5-large")

# train QG model
trainer_qg.train()

```

The corresponding dataset, `lmqg/qg_squad`,⁵ has as columns: `paragraph_answer` (answer-highlighted paragraph), `paragraph_sentence` (sentence-highlighted paragraph), `question` (target question), and `answer` (target answer). The input and the output to the QG model are `paragraph_answer` and `question`, while those to the AE model are `paragraph_sentence` and `answer`. The inputs and the outputs can be specified by passing the name of each column in the dataset to the arguments, `input_types` and `output_types` when instantiating `GridSearcher`.

3.2 QAG Model Generation

In order to generate question-answer pairs from a fine-tuned QAG model, `lmqg` provides the `TransformersQG` class. It takes as input a path to a local model checkpoint or a model name on the HuggingFace Hub in order to generate predictions in a single line of code. The following code snippet shows how to generate a list of question and answer pairs with the fine-tuned QAG model presented in § 2.2. `TransformersQG` decides which model to use for each of AE and QG based on the arguments `model_ae` and `model`.

```

from lmqg import TransformersQG

# instantiate model
model = TransformersQG(
    model="lmqg/t5-base-squad-qg",
    model_ae="lmqg/t5-base-squad-ae"
)

# input paragraph
x = """William Turner was an English
painter who specialised in watercolour
landscapes. One of his best known
pictures is a view of the city of

```

```

Oxford from Hinksey Hill."""

# generation
model.generate_qa(x)
[
    (
        "Who was an English painter
        specialised in watercolour
        landscapes?",
        "William Turner"
    ),
    (
        "Where is William Turner's
        view of Oxford?",
        "Hinksey Hill."
    )
]

```

3.3 QAG Model Evaluation

Similar to other text-to-text generation tasks, we implement an evaluation mechanism that compares the set of generated question-answer pairs $\tilde{Q}_p = \{(\tilde{q}^1, \tilde{a}^1), (\tilde{q}^2, \tilde{a}^2), \dots\}$ to a reference set of gold question-answer pairs $Q_p = \{(q^1, a^1), (q^2, a^2), \dots\}$ given an input paragraph p . Let us define a function to evaluate a single question-answer pair to its reference pair as

$$d_{q,a,\tilde{q},\tilde{a}} = s(t(q, a), t(\tilde{q}, \tilde{a})) \quad (1)$$

$$t(q, a) = \text{"question: \{q\}, answer: \{a\}"} \quad (2)$$

where s is a reference-based metric, and we compute the F_1 score as the final metric as below:

$$F_1 = 2 \frac{R \cdot P}{R + P} \quad (3)$$

$$R = \text{mean} \left(\left[\max_{(q,a) \in Q_c} (d_{q,a,\tilde{q},\tilde{a}}) \right]_{(\tilde{q},\tilde{a}) \in \tilde{Q}_c} \right) \quad (4)$$

$$P = \text{mean} \left(\left[\max_{(\tilde{q},\tilde{a}) \in \tilde{Q}_c} (d_{q,a,\tilde{q},\tilde{a}}) \right]_{(q,a) \in Q_c} \right) \quad (5)$$

Conceptually, the recall (4) and precision (5) computations attempt to “align” each generated question-answer pair to its “most relevant” reference pair. As with traditional precision and recall metrics, precision is aimed at evaluating whether the predicted question-answer pairs are *correct* (or in this case, aligned with the reference question-answer pairs), and recall tests whether there are enough high-quality question-answer pairs. Thus, we refer to the score in (3) as the **QAAligned F1 score**. The quality of the alignment directly depends on the underlying metric s . Furthermore, the complexity of QAAligned is no more than the complexity of the underlying metric, and invariant to the order of generated pairs because of the alignment at computing recall and precision.

⁵https://huggingface.co/datasets/lmqg/qg_squad

Out-of-the-box, `lmqg` implements two variants based on the choice of base_metric s (used for evaluation in § 4): QAAigned BS using BERTScore (Zhang et al., 2019) and QAAigned MS using MoverScore (Zhao et al., 2019). We selected these two metrics as they correlate well with human judgements in QG (Ushio et al., 2022). Nevertheless, the choice of base_metric is flexible and users can employ other natural language generation (NLG) evaluation metrics such as BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), or ROUGE_L (Lin, 2004).

With `lmqg`, QAAigned score can be computed with the QAAignedF1Score class as shown in the code snippet below:

```

from lmqg import QAAignedF1Score

# gold reference and generation
ref = [
    "question: What makes X?, answer: Y",
    "question: Who made X?, answer: Y"]
pred = [
    "question: What makes X?, answer: Y",
    "question: Who build X?, answer: Y",
    "question: When X occurs?, answer: Y"]

# compute QAAigned BS
scorer = QAAignedF1Score(
    base_metric="bertscore")
scorer.get_score(pred, ref)

# compute QAAigned MS
scorer = QAAignedF1Score(
    base_metric="moverscore")
scorer.get_score(pred, ref)

```

4 Evaluation

We rely on the QAG models and datasets included in the library (see § 2). The individual QG components of each model (i.e. the generation of a question given an answer in a paragraph) were extensively evaluated in Ushio et al. (2022). For this evaluation, therefore, we focus on the quality of the predicted questions and answers given a paragraph (i.e. the specific answer is not pre-defined). For each model, we fine-tune, make predictions and compute their QAAigned scores via `lmqg`.

4.1 Results

Monolingual evaluation (English). Table 1 presents the test results on SQuAD for seven English models based on BART, T5 and Flan-T5. The QAG model based on BART_{LARGE} proves to be the best aligned with gold reference question and answers among most of the metrics. As with other

Model	QAAigned BS	QAAigned MS
BART _{BASE}	92.8 / 93.0 / 92.8	64.2 / 64.1 / 64.5
BART _{LARGE}	93.2 / 93.4 / 93.1	64.8 / 64.6 / 65.0
T5 _{SMALL}	92.3 / 92.5 / 92.1	63.8 / 63.8 / 63.9
T5 _{BASE}	92.8 / 92.9 / 92.6	64.4 / 64.4 / 64.5
T5 _{LARGE}	93.0 / 93.1 / 92.8	64.7 / 64.7 / 64.9
Flan-T5 _{SMALL}	92.3 / 92.1 / 92.5	63.8 / 63.8 / 63.8
Flan-T5 _{BASE}	92.6 / 92.5 / 92.8	64.3 / 64.4 / 64.3
Flan-T5 _{LARGE}	92.7 / 92.6 / 92.9	64.6 / 64.7 / 64.5

Table 1: QAAigned scores ($F_1/P/R$) on the test set of SQuAD dataset by different QAG models, where the best score in each metric is shown in boldface.

	Language	QAAigned BS	QAAigned MS
mT5 _{SMALL}	German	81.2 / 80.0 / 82.5	54.3 / 54.0 / 54.6
	Spanish	79.9 / 77.5 / 82.6	54.8 / 53.3 / 56.5
	French	79.7 / 77.6 / 82.1	53.9 / 52.7 / 55.3
	Italian	81.6 / 81.0 / 82.3	55.9 / 55.6 / 56.1
	Japanese	79.8 / 76.8 / 83.1	55.9 / 53.8 / 58.2
	Korean	80.5 / 77.6 / 83.8	83.0 / 79.4 / 87.0
	Russian	77.0 / 73.4 / 81.1	55.5 / 53.2 / 58.3
mT5 _{BASE}	German	76.9 / 76.3 / 77.6	53.0 / 52.9 / 53.1
	Spanish	80.8 / 78.5 / 83.3	55.3 / 53.7 / 57.0
	French	68.6 / 67.6 / 69.7	47.9 / 47.4 / 48.4
	Italian	81.7 / 81.3 / 82.2	55.8 / 55.7 / 56.0
	Japanese	80.3 / 77.1 / 83.9	56.4 / 54.0 / 59.1
	Korean	77.3 / 76.4 / 78.3	77.5 / 76.3 / 79.0
	Russian	77.0 / 73.4 / 81.2	55.6 / 53.3 / 58.4
mBART	German	0 / 0 / 0	0 / 0 / 0
	Spanish	79.3 / 76.8 / 82.0	54.7 / 53.2 / 56.4
	French	75.6 / 74.0 / 77.2	51.8 / 51.0 / 52.5
	Italian	40.1 / 40.4 / 39.9	27.8 / 28.1 / 27.5
	Japanese	76.7 / 74.8 / 78.9	53.6 / 52.3 / 55.1
	Korean	80.6 / 77.7 / 84.0	82.7 / 79.0 / 87.0
	Russian	79.1 / 75.9 / 82.9	56.3 / 54.0 / 58.9

Table 2: QAAigned scores ($F_1/P/R$) on the test set of QG-Bench by different QAG models, where the best score in each language is shown in boldface.

QG experiments and NLP in general, the larger models prove more reliable.

Multilingual evaluation. Table 2 shows the test results of three multilingual models (mBART, mT5_{SMALL} and mT5_{BASE}) in seven languages other than English, using their corresponding language-specific SQuAD-like datasets in QG-Bench for fine-tuning and evaluation.⁶ In this evaluation, no single LM produces the best results across the board, yet QAG models based on mT5_{SMALL} and mT5_{BASE} are generally better than those based on mBART.

⁶The result of mBART in German is zero. Upon further inspection, we found that the fine-tuned answer extraction module did not learn properly, probably due to the limited size of the German dataset. T5 models, however, proved more reliable in this case.

Gold	BART _B	BART _L	T5 _S	T5 _B	T5 _L	Flan-T5 _S	Flan-T5 _B
4.9	4.1	4.2	4.2	4.3	4.3	4.2	4.3

Table 3: Average number of generated question and answer pairs per paragraph on the test set of SQuAD by different QAG models.

Language	Gold	mT5 _{SMALL}	mT5 _{BASE}	mBART
German	4.6	10.1	8.4	0.0
Spanish	1.3	4.6	4.8	4.7
French	1.3	4.9	3.6	5.4
Italian	3.8	4.7	4.6	2.5
Japanese	1.3	6.6	6.8	3.6
Korean	1.3	6.7	6.3	6.7
Russian	1.3	4.8	4.9	4.7

Table 4: The averaged number of generated question and answer pairs per paragraph on the test set of QG-Bench for each language.

4.2 Number of Generated Questions and Answers

Table 3 and Table 4 show the averaged number of generated question-answer pairs and compare it to the number in the gold dataset. For English, there is a small difference across all QAG models, with all generating fewer pairs than the gold dataset, but with a limited margin. For other languages, however, there are clear differences across QAG models, with the numbers of question-answer pairs generated by the QAG models always being larger than those in the gold dataset. When comparing the number of pairs generated by the QAG models with their QAAI scores, in languages such as German, Spanish, and Korean, QAG models that generated a larger number question-answer pairs achieved higher scores, not only recall-wise but also generally for F1.

5 AutoQG

Finally, we present AutoQG (<https://autoqg.net>), an online QAG demo where users can generate question-answer pairs for texts in eight languages (English, German, Spanish, French, Italian, Japanese, Korean, Russian) by simply providing a context document. We deploy the QAG models described in § 2. In addition to the features described above, the online demo shows perplexity computed via `lmpp1`,⁷ a Python library to compute perplexity given any LM architecture. This feature helps us provide a ranked list of generation to the user. Although we can compute perplexity for non-English

⁷<https://pypi.org/project/lmpp1>

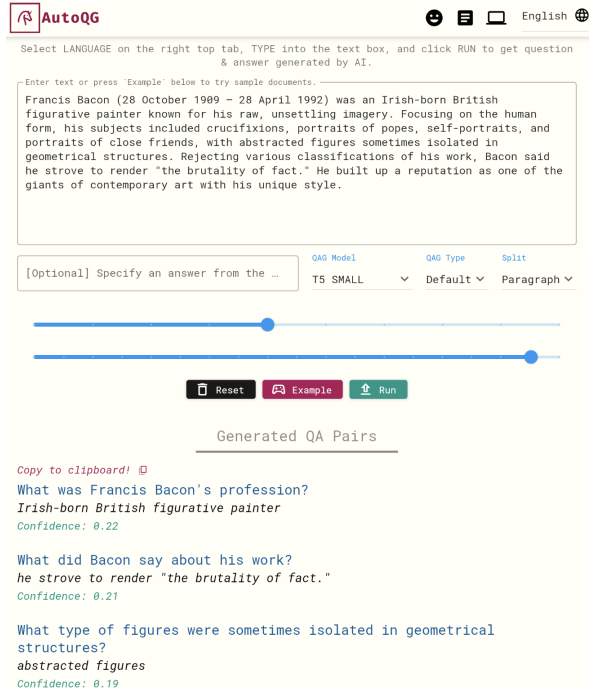


Figure 2: A screenshot of AutoQG with an example of question and answer generation over a paragraph.

generations based on the QAG models in each language, it entails large memory requirements on the the hosting server. As such, we compute a lexical overlap between the question and the document as a computationally-light alternative to the perplexity, which is defined as:

$$1 - \frac{|q \cap p|}{|q|} \quad (6)$$

where $|\cdot|$ is the number of characters in a string, and $q \cap p$ is the longest sub-string of the question q matched to the paragraph p .

Figure 2 and Figure 3 show examples of the interface with English and Japanese QAG, where there is a tab to select QAG models, language, and parameters at generation including the beam size and the value for nucleus sampling (Holtzman et al., 2020). Optionally, users can specify an answer and generate a single question on it with the QG model, as shown in Figure 4. A short introduction video to AutoQG is available at <https://youtu.be/T6G-D9JtYyc>.

6 Conclusion

In this paper, we introduced `lmqg`, a Python package to fine-tune, evaluate and deploy QAG models with a few lines of code. The library implements the QAG task as an efficient integration of answer



Figure 3: A screenshot of AutoQG with an example of question and answer generation over a paragraph in Japanese.



Figure 4: A screenshot of AutoQG when an answer is specified by the user.

extraction and question generation, and includes automatic reference-based metrics for model evaluation. Finally, we showcase AutoQG, an online demo where end users can benefit from QAG models without any programming knowledge. AutoQG enables the selection of features going from different models and languages to controlling the diversity of the generation.

Limitations

The focus on this paper was introducing software to make QAG models available to as many practitioners as possible, but there are a couple of limitations in the models and evaluation metrics we proposed.

First, our released QAG models assume a paragraph up to around 500 tokens as an input, and longer documents can not be directly fed into the models. Additionally, the released QAG models were fine-tuned on questions that require one-hop reasoning only, so they are unable to generate multi-hop reasoning.

Second, the QAAligned score is a framework to extend any NLG metric to match the prediction to the reference when they are different in size, where we employed two well-established metrics (BERTScore and MoverScore) as underlying metrics. Since those underlying metrics are already proven to be effective (Zhang et al., 2019; Zhao et al., 2019; Ushio et al., 2022), we have not conducted any human annotation for QAG specifically. Nonetheless, an extended human evaluation could help provide more insights on other limitations of the model not detected by the automatic evaluation.

Ethics Statement

While the QAG models are fine-tuned on pre-trained language models, which are known to contain some toxic contents (Schick et al., 2021), an internal check does not reveal any toxic generation. However, there is a potential risk that the QAG model could generate toxic text due to the underlying LMs.

References

- Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021. [Improving question answering model robustness with synthetic adversarial data generation](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Johannes Bjerva, Nikita Bhutani, Behzad Golshan, Wang-Chiew Tan, and Isabelle Augenstein. 2020. [SubjQA: A Dataset for Subjectivity and Review Comprehension](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5480–5494, Online. Association for Computational Linguistics.
- Carrino Casimiro Pio, Costa-jussa Marta R., and Fonollosa Jose A. R. 2019. [Automatic Spanish Translation](#)

- of the SQuAD Dataset for Multilingual Question Answering. *arXiv e-prints*, page arXiv:1912.05200v1.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Danilo Croce, Alexandra Zelenanska, and Roberto Basili. 2018. Neural learning for question answering in italian. In *AI*IA 2018 – Advances in Artificial Intelligence*, pages 389–402, Cham. Springer International Publishing.
- Michael Denkowski and Alon Lavie. 2014. **Meteor universal: Language specific translation evaluation for any target language**. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.
- Martin d’Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. 2020. **FQuAD: French question answering dataset**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online. Association for Computational Linguistics.
- Pavel Efimov, Andrey Chertok, Leonid Boytsov, and Pavel Braslavski. 2020. Sberquad–russian reading comprehension dataset: Description and analysis. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 3–15. Springer.
- Michael Heilman and Noah A. Smith. 2010. **Good question! statistical ranking for question generation**. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. **The curious case of neural text de-generation**. In *International Conference on Learning Representations*.
- 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.
- Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. **Unsupervised question answering by cloze translation**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.
- Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. **PAQ: 65 million probably-asked questions and what you can do with them**. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.
- Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1.0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.
- 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.
- David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. **Generating natural language questions to support learning on-line**. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria. 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.
- John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR.
- Timo Möller, Julian Risch, and Malte Pietsch. 2021. **Germanquad and germandpr: Improving non-english question answering and passage retrieval**.
- Nedjma Ousidhoum, Zhangdie Yuan, and Andreas Vlachos. 2022. **Varifocal question generation for fact-checking**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2532–2544, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- 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.
- Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. **Training question answering models from synthetic data**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.
- Valentina Pyatkin, Paul Roit, Julian Michael, Yoav Goldberg, Reut Tsarfaty, and Ido Dagan. 2021. **Asking it all: Generating contextualized questions for any**

semantic role. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1429–1441, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

ByungHoon So, Kyuhong Byun, Kyungwon Kang, and Seongjin Cho. 2022. Jaquad: Japanese question answering dataset for machine reading comprehension. *arXiv preprint arXiv:2202.01764*.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2022. Generative language models for paragraph-level question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, U.A.E. Association for Computational Linguistics.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. An empirical comparison of lm-based question and answer generation methods. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada. 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.

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.

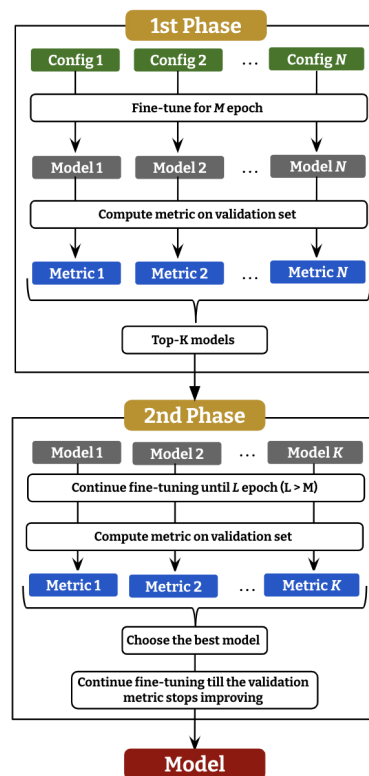


Figure 5: An overview of the hyper-parameter search implemented as GridSearcher.

Shiyue Zhang and Mohit Bansal. 2019. **Addressing semantic drift in question generation for semi-supervised question answering.** 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 2495–2509, Hong Kong, China. Association for Computational Linguistics.

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*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. **MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance.** 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 563–578, Hong Kong, China. Association for Computational Linguistics.

A Grid Search

To fine-tune LMs on QAG, one can use the GridSearcher class of lmq, which performs LM fine-tuning with a two-stage optimization of hyper-parameter, a set of parameters to be used at fine-tuning such as learning rate or batch size, as de-

scribed in Figure 5. Let us assume that we want to find an optimal combination of the learning rate and random seed from a list of candidates [1e-4,1e-5] and [0,1] for learning rate and random seed respectively on QG as an example. We also assume a training and a validation dataset to train a model on the task and an evaluation score that reflects a performance of a model (eg. BLEU4(Papineni et al., 2002)), and we define a search-space as a set including all the combinations of those candidates, i.e. {(1e-4, 0), (1e-4, 1), (1e-5, 0), (1e-5, 1)}. The goal of the GridSearcher is to find the best combination to train a model on the training dataset for the target task over the search-space with respect to the evaluation score computed on the validation dataset.

Brute-force approach such as to train model over every combination in the search-space can be a highly-inefficient, so GridSearcher employs a two-stage search method to avoid training for all the combinations, while being able to reach to the optimal combination as possible. To be precise, given an epoch size L (epoch), the first stage fine-tunes all the combinations over the search-space, and pauses fine-tuning at epoch M (epoch_partial). The top- K combinations (n_max_config) are then selected based on the evaluation score computed over the validation dataset, and they are resumed to be fine-tuned until the last epoch. Once the K chosen models are fine-tuned at second stage, the best model is selected based on the evaluation score, which is kept being fine-tuned until the evaluation score decreases.

The dataset for training and validation can be any datasets shared in the HuggingFace Hub, and one can specify the input and the output to the model from the column of the dataset by the arguments input_types and output_types at instantiating GridSearcher. For example, the following code shows how we can fine-tune T5 (Raffel et al., 2020) on question generation, a sub-task of QAG, with SQuAD (Rajpurkar et al., 2016), where the dataset lmqg/qg_squad is shared at https://huggingface.co/datasets/lmqg/qg_squad on the HuggingFace Hub, which has columns of paragraph_answer, that contains a answer-highlighted paragraph, and question, which is a question corresponding to the answer highlighted in the paragraph_answer. We choose them as the input and the output to the model respectively by passing the name of each column to the arguments, input_types and

output_types.

```
from lmqg import GridSearcher

# instantiate the trainer
trainer = GridSearcher(
    dataset_path="lmqg/qg_squad",
    input_types="paragraph_answer",
    output_types="question",
    model="t5-large",
    batch_size=128,
    epoch=10,
    epoch_partial=2,
    n_max_config=3,
    lr=[1e-4,1e-5],
    random_seed=[0,1])

# train model
trainer.train()
```

OpenSLU: A Unified, Modularized, and Extensible Toolkit for Spoken Language Understanding

Libo Qin^{1*}, Qiguang Chen^{2*}, Xiao Xu², Yunlong Feng², Wanxiang Che²

¹School of Computer Science and Engineering

¹Central South University, China

²Research Center for Social Computing and Information Retrieval

²Harbin Institute of Technology, China

lbqin@csu.edu.cn, {qgchen, xxu, ylfeng, car}@ir.hit.edu.cn

Abstract

Spoken Language Understanding (SLU) is one of the core components of a task-oriented dialogue system, which aims to extract the semantic meaning of user queries (e.g., intents and slots). In this work, we introduce OpenSLU, an open-source toolkit to provide a unified, modularized, and extensible toolkit for spoken language understanding. Specifically, OpenSLU unifies 10 SLU models for both single-intent and multi-intent scenarios, which support both non-pretrained and pretrained models simultaneously. Additionally, OpenSLU is highly modularized and extensible by decomposing the model architecture, inference, and learning process into reusable modules, which allows researchers to quickly set up SLU experiments with highly flexible configurations. OpenSLU is implemented based on PyTorch, and released at <https://github.com/LightChen233/OpenSLU>.

1 Introduction

Spoken Language Understanding (SLU), which is used to extract the semantic frame of user queries (e.g., intents and slots) (Tur and De Mori, 2011). Typically, SLU consists of two sub-tasks: intent detection and slot filling. Take the utterance shown in Figure 1 as an example, given “Listen to Rock Music”, the outputs include an intent class label (i.e., Listen-to-Music) and a slot label sequence (i.e., O, O, B-music-type, I-music-type).

Since intent detection and slot filling are highly tied (Qin et al., 2021c), dominant methods in the literature explore joint models for SLU to capture shared knowledge (Goo et al., 2018; Wang et al., 2018; Qin et al., 2019). Recently, Gangadhariah and Narayanaswamy (2019) shows that, in the Amazon internal dataset, 52% of examples contain multiple intents. Inspired by this observation, various SLU works shift their eye from single-intent SLU to multi-intent SLU scenario (Gangadhariah and Narayanaswamy, 2019; Qin et al., 2020b; Casanueva et al., 2022; Moghe et al., 2022).

*Equal Contribution

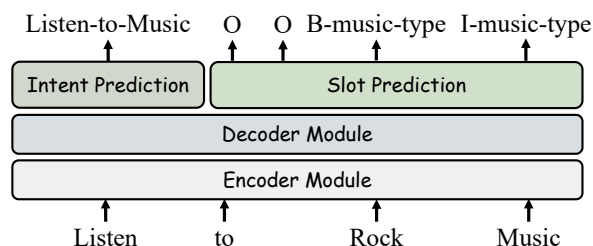


Figure 1: An example of spoken language understanding. Listen-to-Music stands for the intent label while {O, O, B-music-type, I-music-type} denotes the slot sequence labels.

iah and Narayanaswamy, 2019; Qin et al., 2020b; Casanueva et al., 2022; Moghe et al., 2022).

Thanks to the development of neural network, especially the successful use of large pretrained models, remarkable success have been witnessed in SLU. Nevertheless, there still lacks a unified open-source framework to facilitate the SLU community. In this work, we make the first attempt to introduce OpenSLU, a unified, modularized, and extensible toolkit for SLU, which aims to help researchers to set up experiments and develop their new models quickly. The main features of OpenSLU are:

- **Unified and modularized toolkit.** OpenSLU is the first unified toolkit to support both single-intent and multi-intent SLU scenarios. Meanwhile, it is highly modularized by decoupling SLU models into a set of highly reusable modules, including data module, model module, evaluation module, as well as various common components and functions. Such modularization allows users to quickly reimplement SLU baselines or develop their new SLU models by re-using provided modules or adding new modules.
- **Extensible and flexible toolkit.** OpenSLU is configured by configuration objects, which is extensible and can be initialized from YAML files. This enables users can easily develop

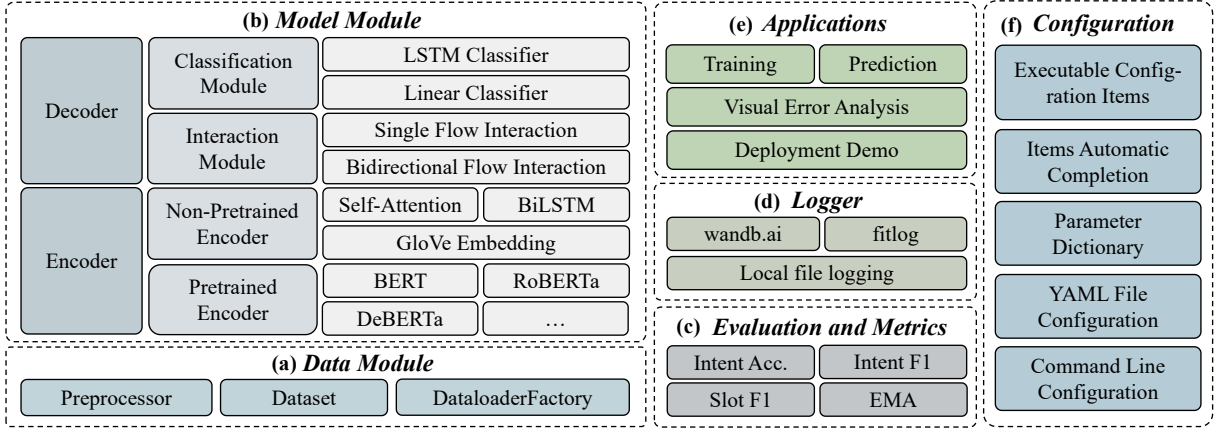


Figure 2: An overall workflow of OpenSLU, which consists of (a) Data Module, (b) Model Module, (c) Evaluation and Metrics, (d) Logger, (e) Applications and (f) Configuration.

their models by simply extending the configurations. Additionally, we provide various interfaces of various common functions or modules in SLU models, including Encoder and Decoder module. Besides, the interfaces of our toolkit are fully compatible with the PyTorch interface, which allows seamless integration and flexibly rewriting any sub-module in the toolkit.

- **Visualization Tool.** We provide a visualization tool to help users to view all errors of the model directly. With the help of visualization tool, we can get a clearer picture: where we are and where we should focus our efforts to improve the performance of the model, which helps to develop a more superior framework.

To our knowledge, this is the first unified, modularized, and extensible toolkit for SLU. We hope our work can help researchers to quickly initiate experiments and spur more breakthroughs in SLU¹.

2 Architecture and Design

Figure 2 illustrates the overall workflow of OpenSLU. In this section, we describe the (a) Data Module (§2.1); (b) Model Module (§2.2); (c) Evaluation and Metrics (§2.3) and other common modules (Logger, Applications and Configuration module) (§2.4).

2.1 Data Module

OpenSLU offers an integrated data format in the data module (see Figure 2(a)) for

¹Video introduction about OpenSLU is available at https://youtu.be/uOXh47m_xhU.

SLU models, which can be denoted as: $raw\ text \rightarrow Preprocessor \rightarrow Dataset \rightarrow DataLoaderFactory \rightarrow model\ input$.

Given the input *raw text*, *Preprocessor* sub-module first pre-process different raw texts to an integrated *.jsonl* format that contains slot, text and intent, which is formatted as:

```
{
  "slot": [List of Slot Value],
  "text": [List of Text],
  "intent": [Intent Value]
}
```

The Dataset sub-module offers a range of data processing operations to support both pretrained and non-pretrained models. For pretrained models, these operations include lowercase conversion, BPE-tokenization, and slot alignment, while for non-pretrained models, the sub-module handles word-tokenization and vocabulary construction.

Finally, *DataLoaderFactory* sub-model is used for creating *DataLoader* to manage the data stream for models.

2.2 Model Module

As shown in Figure 2(b), the overall model module contains encoder module (§2.2.1) and decoder module (§2.2.2).

2.2.1 Encoder

For the encoder module, we implement both non-pretrained models and pretrained models. In non-pretrained models, we offer the widely used SLU encoders including self-attentive (Vaswani et al., 2017; Qin et al., 2019) and BiLSTM (Hochreiter and Schmidhuber, 1997; Goo et al., 2018; Liu et al.,

2020b) encoder. Additionally, we support auto-load GloVe embedding (Pennington et al., 2014).

In pretrained models, OpenSLU supports various encoders including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020a), ELECTRA (Clark et al., 2020), DeBERTa_{v3} (He et al., 2021).

2.2.2 Decoder

Since slot filling and intent detection are highly related, dominant methods in the literature employ joint models to capture the shared knowledge across the related tasks (Goo et al., 2018; Wang et al., 2018; Chen et al., 2019). To support the joint modeling paradigm, the decoder in OpenSLU contains two sub-modules: (1) interaction module for capturing interaction knowledge for slot filling and intent detection and (2) classification module for the final prediction results.

Interaction Module. As summarized in Qin et al. (2021c), the interaction module consists of two widely used interaction types, including *single flow interaction* and *bidirectional flow interaction*.

- **Single Flow Interaction** refers to the flow of information from intent to slot in one direction as illustrated in Figure 3(a). A series of studies (Goo et al., 2018; Li et al., 2018; Qin et al., 2019) have achieved remarkable improvements in performance by guiding slot filling with intent detection information.
- **Bidirectional Flow Interaction** stands for the bidirectional cross-impact between intent detection and slot filling can be considered, which is shown in Figure 3(b). Another series of works (Wang et al., 2018; E et al., 2019; Liu et al., 2019; Qin et al., 2021a) build the bidirectional connection across slot filling and intent detection to enhance each other.

Based on the two types of interaction, users can easily design the interaction module and interaction order via our provided classic interaction modules and customized configurations.

Classification Module. It aims to transform hidden states after the interaction module into final classification logits. There are two types of classification modules supported by OpenSLU:

- **MLP Classifier.** Multi-Layer Perceptron (MLP) Classifier is a fundamental classification decoding algorithm. Nevertheless, the method ignores the dependency across tokens.

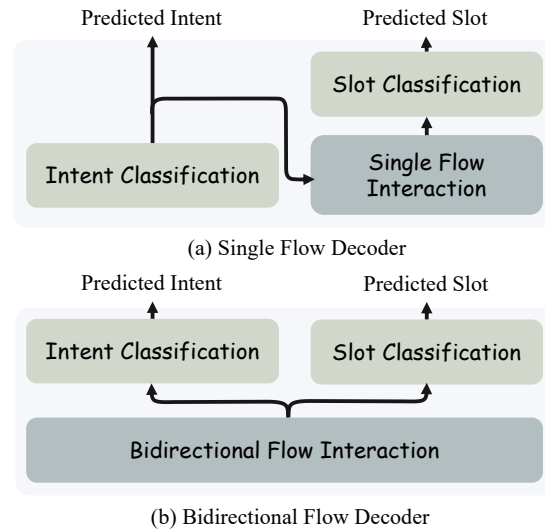


Figure 3: A brief illustration of Single Flow Decoder (a) vs. Bidirectional Flow Decoder (b).

- **LSTM Classifier.** It indicates that we adopt an LSTM classifier for the final prediction, which has the advantage of modeling the dependency of tokens (from left to right). However, it is an autoregressive classification module for SLU, which cannot be parallel to speed up the decoding prediction.

To improve the quality of SLU prediction results, we also implement several SLU tricks, like teacher-forcing and token-level intent detection (Qin et al., 2019). Users can switch between different prediction strategies by simply setting up the hyperparameter to improve performance.

2.3 Evaluation and Metrics

Following Goo et al. (2018); Qin et al. (2021c), we support various metrics for SLU (shown in Figure 2(c)), including Slot F1 Score, Intent Accuracy, Intent F1, and Exactly Match Accuracy (EMA).

- **Slot F1 Score** (Goo et al., 2018; Qin et al., 2019) is used for assessing slot filling performance. This metric is calculated as the harmonic mean between precision and recall.
- **Intent Accuracy** (Goo et al., 2018; Qin et al., 2019) is a measure used to evaluate the accuracy of intent detection, based on the ratio of correctly predicted intents.
- **Intent F1 Score** (Gangadharaiah and Narayanaswamy, 2019; Qin et al., 2020b) is adopted to evaluate the macro F1 Score of the predicted intents in the multi-intent detection.

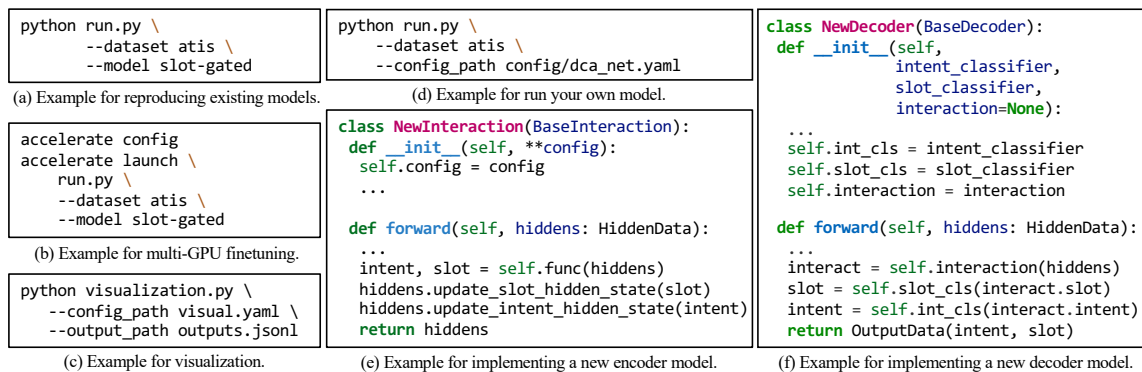


Figure 4: Example usage of OpenSLU.

- **Exact Match Accuracy** (Goo et al., 2018; Qin et al., 2019, 2020b) takes intent detection as well as slot filling into account simultaneously. This metric is calculated as the ratio of sentences for which both the intent and slot are predicted correctly within a sentence.

2.4 Common Modules

Logger. We provide a generic Logger component to help users to track the process of model building including wandb.ai, fitlog and local file logging (see Figure 2(d)).

Applications. We provide complete scripts in the Application (see Figure 2(e)) for training, prediction, visual error analysis, and the final stage of model deployment.

Configuration. As shown in Figure 2(f), our toolkit employs Configuration module to manage the model configuration, training parameters, and training and analysis data. We will introduce more details in Section Toolkit Usage (§3).

3 Toolkit Usage

3.1 Reproducing Existing Models

For reproducing an existing model implemented by OpenSLU on different datasets, users are required only to specify the dataset and model by setting hyper-parameters, i.e., *model* and *dataset*. Experiments can be reproduced in a simple command line instruction, as shown in Figure 4 (a). This instruction aims to fine-tuning Slot-Gated (Goo et al., 2018) model on ATIS (Hemphill et al., 1990) dataset. With YAML configuration files, we can modify hyper-parameters conveniently, which allows users can reproduce various experiments quickly without modifying the source code. In

addition, we designed OpenSLU to work on a variety of hardware platforms. If the hyper-parameter *device* is set to “*cuda*”, CUDA devices will be used. Otherwise, CPU will be employed by default. As shown in Figure 4 (b), we also support distributed training on multi-GPU by setting hyper-parameters and command line parameters.

3.2 Customizable Combination Existing Components

As the model is designed as reusable modules, users can easily reuse modules via the call of interface or configuration files. More specifically, for the interface, users can call common-used encoder and decoder modules in one line of code from the pre-configured library. For configuration files, users can combine existing component libraries only through configuration files, thus creating a customized model.

It can be useful for users in cross-cutting areas, such as biology, that are unfamiliar with using Python code to create models, as it allows them to create their own models without using any Python code. Such features can potentially make it easier to build and test models more rapidly. Similarly, the customized model can be trained by specifying the relevant configuration file path and running simple command line instructions, as shown in Figure 4(d).

3.3 Implementing a New SLU Model

Since OpenSLU split the model into fine-grained components, users can directly reuse modules through configuration files. Specifically, when users aim to implement a new SLU model, only a few key innovative modules need to be rewritten by users, including a specific Model class and 2 functions as follows:

Model	ATIS			SNIPS		
	Slot F1.(%)	Intent Acc.(%)	EMA(%)	Slot F1.(%)	Intent Acc.(%)	EMA(%)
<i>Non-Pretrained Models</i>						
Slot Gated (Goo et al., 2018)	94.7	94.5	82.5	93.2	97.6	85.1
Bi-Model (Wang et al., 2018)	95.2	96.2	85.6	93.1	97.6	84.1
Stack Propagation (Qin et al., 2019)	95.4	96.9	85.9	94.6	97.9	87.1
DCA Net (Qin et al., 2021a)	95.9	97.3	87.6	94.3	98.1	87.3
<i>Pretrained Models</i>						
Joint BERT (Chen et al., 2019)	95.8	97.9	88.6	96.4	98.4	91.9
RoBERTa (Liu et al., 2020a)	95.8	97.8	88.1	95.7	98.1	90.6
ELECTRA (Clark et al., 2020)	95.8	96.9	87.1	95.7	98.3	90.1
DeBERTa _{v3} (He et al., 2021)	95.8	97.8	88.4	97.0	98.4	92.7

Table 1: Main results of single-intent SLU. All baseline results are re-implemented by OpenSLU.

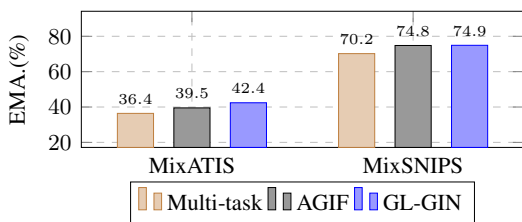


Figure 5: Main results of multi-intent SLU on EMA. All baseline results are re-implemented by OpenSLU.

- `__init__()` function. This function aims for parameter initialization, global variable definition, and so on. All modules can be inserted into the system by configuring the `__model_target__` hyper-parameters, so as to quickly and automatically build the model.
- `forward()` function. This function mainly focuses on forward data flow and learning the parameters according to the pre-defined configuration.

In most cases, rewriting Interaction module is enough for building a new SLU model. As shown in Figure 4(e), this module accepts `HiddenData` data object as input and returns with `HiddenData` data object. `HiddenData` contains the `hidden_states` for intent and slot, and other helpful information. With the advancement of SLU research, patterns of decoders become increasingly complex (Xing and Tsang, 2022; Cheng et al., 2022). Therefore, to further meet the needs of complex exploration, we provide the `BaseDecoder` class, and the user can simply override the `forward()` function in class, which accepts `HiddenData` as input data format and `OutputData` as output data format, as shown in Figure 4(f).

4 Experiments

Extensive reproduction experiments are conducted to evaluate the effectiveness of OpenSLU.

4.1 Data Settings

In single-intent SLU, we employ two widely used benchmarks including ATIS (Hemphill et al., 1990) and SNIPS dataset (Coucke et al., 2018).

In multi-intent SLU scenario, we support 2 widely used datasets: MixATIS and MixSNIPS (Qin et al., 2020b), which are collected from the ATIS, SNIPS by simple conjunctions, e.g., “and”, to connect sentences with different intents.

4.2 Result Reproduction

We implement various state-of-the-art SLU models. For single-intent SLU methods, we re-implement the following baselines: (1) Slot Gated (Goo et al., 2018); (2) Bi-Model (Wang et al., 2018); (3) Stack Propagation (Qin et al., 2019); (4) DCA Net (Qin et al., 2021a); (5) Joint Bert (Chen et al., 2019); (6) RoBERTa (Liu et al., 2020a); (7) ELECTRA (Clark et al., 2020); (8) DeBERTa_{v3} (He et al., 2021). For multi-intent SLU methods, we adopt the following baselines: (1) AGIF (Qin et al., 2020b); (2) GL-GIN (Qin et al., 2021b).

The reproduction results are illustrated in Table 1, we observe that OpenSLU toolkit can reproduce the comparable results reported in previous works, which verify the effectiveness of OpenSLU. In addition, OpenSLU can outperform some reported results in previous published work, which further shows the superiority of OpenSLU. Meanwhile, the same trend can be observed in multi-intent SLU setting, which is shown in Figure 5.

4.3 Visualization Analysis

According to a number of studies (Vilar et al., 2006; Wu et al., 2019; Ribeiro et al., 2020; Paleyes et al., 2022), model metrics tests alone no longer adequately reflect the model’s performance. To help researchers further improve their models, we pro-

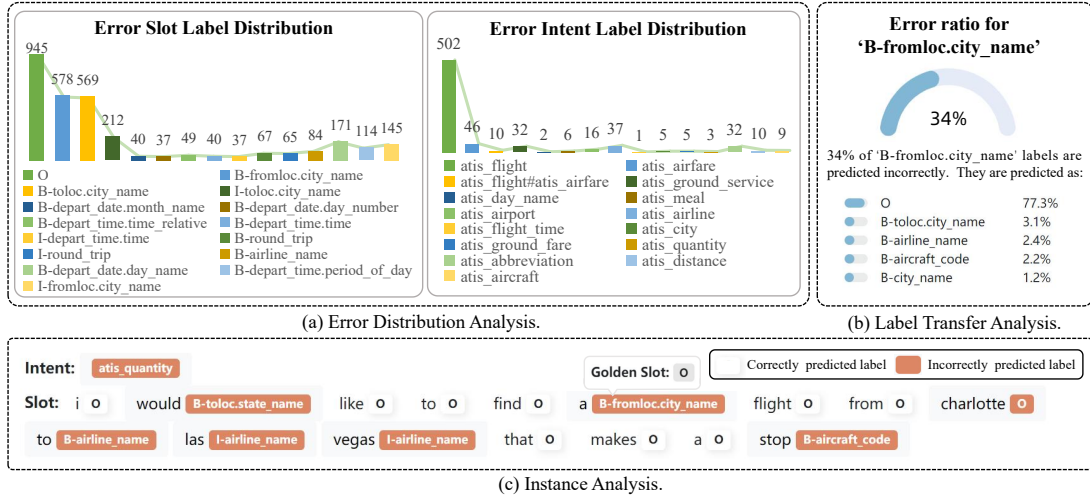


Figure 6: Visual analytics consists of three main functions including Error Distribution Analysis (a), Label Transfer Analysis (b) and Instance Analysis (c).

vide a tool for visual error analysis including three main parts: (a) error distribution analysis; (b) label transfer analysis; and (c) instance analysis (see Figure 6). And the visual analysis interface can be run with the command as shown in the Figure 4(c).

4.3.1 Error Distribution Analysis.

We provide error distribution analysis that presents the number and percentage of label errors predicted by the model. By viewing the error distributions, the model can be easily analyzed and studied qualitatively (Caubrière et al., 2020). As a result, the weaknesses of each system can be better understood and improvements can be made to the model in the future.

Take the error in Figure 6(a) as an example, a large number of 'atis_flight' labels are incorrectly predicted compared with all other labels. Therefore, we should pay more attention on how to improve the performance of 'atis_flight' labels.

4.3.2 Label Transfer Analysis.

Label Transfer Analysis module first offers the percentage of incorrect predictions for each label and provides the probability of being incorrectly predicted as each of the other labels to present a fine-grained statistics for a better understanding of issues such as invisible bias in the model (Wu et al., 2019; Ribeiro et al., 2020).

For example, Figure 6(b) shows the details in incorrect prediction on 'B-fromloc.city_name'. We observe 34% of 'B-fromloc.city_name' predict incorrectly and 77.3% of error labels are predicted as 'O'. By having access to this information,

users can be better guided to improve their data or label learning methods to prevent those error predictions.

4.3.3 Instance Analysis.

In order to provide a better case study, OpenSLU offers a instance-level analysis view by highlighting error results and interactively checking all golden labels (shown in Figure 6(c)). Such instance analysis allows users to examine data on a case-by-case basis in an intuitive way. This can be seen easily in Figure 6(c), where token 'a' is predicted as 'B-fromloc.city_name' instead of 'O'.

Furthermore, we also deploy OpenSLU into the Gradio² platform, which allows users to connect the demo directly to the public network and access it via the computer or mobile device.

5 Conclusion

This paper introduces OpenSLU, a unified, modularized, and extensible toolkit for spoken language understanding. In our toolkit, we implement 10 models on both single- and multi-intent SLU settings, both covering the categories of non-pretrained and pretrained language models. Our toolkit can be easily applied to other SLU settings, which is extensible to support seamless incorporation of other external modules. To the best of our knowledge, this is the first open-resource toolkit for SLU and we hope OpenSLU can attract more breakthroughs in SLU. In the future, we can extend OpenSLU to support cross-lingual (Qin et al., 2020a; Zheng et al., 2022) and profile (Xu et al., 2022) SLU scenario.

²<https://www.gradio.app>

Acknowledgements

This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 62236004 and 61976072. Libo Qin is the corresponding author.

References

- Inigo Casanueva, Ivan Vulić, Georgios Spithourakis, and Paweł Budzianowski. 2022. [NLU++: A multi-label, slot-rich, generalisable dataset for natural language understanding in task-oriented dialogue](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1998–2013, Seattle, United States. Association for Computational Linguistics.
- Antoine Caubrière, Sahar Ghannay, Natalia Tomashenko, Renato De Mori, Antoine Laurent, Emmanuel Morin, and Yannick Estève. 2020. Error analysis applied to end-to-end spoken language understanding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8514–8518. IEEE.
- Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.
- Lizhi Cheng, Wenmian Yang, and Weijia Jia. 2022. A scope sensitive and result attentive model for multi-intent spoken language understanding. *arXiv preprint arXiv:2211.12220*.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [ELECTRA: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, 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*.
- 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.
- Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. [A novel bi-directional interrelated model for joint intent detection and slot filling](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471, Florence, Italy. Association for Computational Linguistics.
- Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2019. [Joint multiple intent detection and slot labeling for goal-oriented dialog](#). 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 564–569, Minneapolis, Minnesota. 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.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTaV3: Improving DeBERTa using Electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.
- 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*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long Short-Term Memory](#). *Neural Computation*, 9(8):1735–1780.
- Changliang Li, Liang Li, and Ji Qi. 2018. [A self-attentive model with gate mechanism for spoken language understanding](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833, Brussels, Belgium. Association for Computational Linguistics.
- Yijin Liu, Fandong Meng, Jinchao Zhang, Jie Zhou, Yufeng Chen, and Jinan Xu. 2019. [CM-net: A novel collaborative memory network for spoken language understanding](#). 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 1051–1060, Hong Kong, China. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. [RoBERTa: A robustly optimized BERT pretraining approach](#).
- Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020b. [Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems](#). *Proceedings of the AAAI*

- Conference on Artificial Intelligence*, 34(05):8433–8440.
- Nikita Moghe, Evgeniia Razumovskaia, Liane Guillou, Ivan Vulić, Anna Korhonen, and Alexandra Birch. 2022. Multi3nlu++: A multilingual, multi-intent, multi-domain dataset for natural language understanding in task-oriented dialogue. *arXiv preprint arXiv:2212.10455*.
- Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. 2022. **Challenges in deploying machine learning: A survey of case studies**. *ACM Comput. Surv.*, 55(6).
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. **GloVe: Global vectors for word representation**. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. **A stack-propagation framework with token-level intent detection for spoken language understanding**. 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 2078–2087, Hong Kong, China. Association for Computational Linguistics.
- Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. 2021a. **A co-interactive transformer for joint slot filling and intent detection**. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8193–8197.
- Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020a. **Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp**. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3853–3860. International Joint Conferences on Artificial Intelligence Organization. Main track.
- Libo Qin, Fuxuan Wei, Tianbao Xie, Xiao Xu, Wanxiang Che, and Ting Liu. 2021b. **GL-GIN: Fast and accurate non-autoregressive model for joint multiple intent detection and slot filling**. 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 178–188, Online. Association for Computational Linguistics.
- Libo Qin, Tianbao Xie, Wanxiang Che, and Ting Liu. 2021c. **A survey on spoken language understanding: Recent advances and new frontiers**. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4577–4584. International Joint Conferences on Artificial Intelligence Organization. Survey Track.
- Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020b. **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, Online. Association for Computational Linguistics.
- Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. **Beyond accuracy: Behavioral testing of NLP models with CheckList**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.
- Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.
- 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.
- David Vilar, Jia Xu, L. F. D’Haro, and Hermann Ney. 2006. **Error analysis of statistical machine translation output**. In *International Conference on Language Resources and Evaluation*, pages 697–702.
- Yu Wang, Yilin Shen, and Hongxia Jin. 2018. **A bi-model based RNN semantic frame parsing model for intent detection and slot filling**. 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 309–314, New Orleans, Louisiana. Association for Computational Linguistics.
- Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. **Errudite: Scalable, reproducible, and testable error analysis**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy. Association for Computational Linguistics.
- Bowen Xing and Ivor Tsang. 2022. **Co-guiding net: Achieving mutual guidances between multiple intent detection and slot filling via heterogeneous semantics-label graphs**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 159–169, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Xiao Xu, Libo Qin, Kaiji Chen, Guoxing Wu, Linlin Li, and Wanxiang Che. 2022. **Text is no more enough! a benchmark for profile-based spoken language understanding**. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11575–11585.
- Bo Zheng, Zhouyang Li, Fuxuan Wei, Qiguang Chen, Libo Qin, and Wanxiang Che. 2022. **HIT-SCIR at MMNLU-22: Consistency regularization for multilingual spoken language understanding**. In *Proceedings of the Massively Multilingual Natural Language Understanding Workshop (MMNLU-22)*.

SanskritShala: A Neural Sanskrit NLP Toolkit with Web-Based Interface for Pedagogical and Annotation Purposes

Jivnesh Sandhan¹, Anshul Agarwal¹, Laxmidhar Behera^{1,3},

Tushar Sandhan¹ and Pawan Goyal²

¹IIT Kanpur, ²IIT Kharagpur, ³IIT Mandi

jivnesh@iitk.ac.in, pawang@cse.iitkgp.ac.in

Abstract

We present a neural Sanskrit Natural Language Processing (NLP) toolkit named SanskritShala¹ to facilitate computational linguistic analyses for several tasks such as word segmentation, morphological tagging, dependency parsing, and compound type identification. Our systems currently report state-of-the-art performance on available benchmark datasets for all tasks. SanskritShala is deployed as a web-based application, which allows a user to get real-time analysis for the given input. It is built with easy-to-use interactive data annotation features that allow annotators to correct the system predictions when it makes mistakes. We publicly release the source codes of the 4 modules included in the toolkit, 7 word embedding models that have been trained on publicly available Sanskrit corpora and multiple annotated datasets such as word similarity, relatedness, categorization, analogy prediction to assess intrinsic properties of word embeddings. So far as we know, this is the first neural-based Sanskrit NLP toolkit that has a web-based interface and a number of NLP modules. We are sure that the people who are willing to work with Sanskrit will find it useful for pedagogical and annotative purposes. SanskritShala is available at: <https://cnerg.iitkgp.ac.in/sanskritshala>. The demo video of our platform can be accessed at: <https://youtu.be/x0X31Y9k0mw4>.

1 Introduction

Sanskrit is a culture-bearing and knowledge-preserving language of ancient India. Digitization has come a long way, making it easy for people to access ancient Sanskrit manuscripts (Goyal et al., 2012; Adiga et al., 2021). However, we find that the utility of these digitized manuscripts is limited due to the user’s lack of language expertise and various linguistic phenomena exhibited by the language. This motivates us to investigate how we

can utilize natural language technologies to make Sanskrit texts more accessible.

The aim of this research is to create neural-based Sanskrit NLP systems that are accessible through a user-friendly web interface. The Sanskrit language presents a range of challenges for building deep learning solutions, such as the *sandhi* phenomenon, a rich morphology, frequent compounding, flexible word order, and limited resources (Sandhan et al., 2022d; Krishna et al., 2021; Sandhan et al., 2021, 2019). To overcome these challenges, 4 preliminary tasks were identified as essential for processing Sanskrit texts: word segmentation, morphological tagging, dependency parsing, and compound type identification. The word segmentation task is complicated by the *sandhi* phenomenon, which transforms the word boundaries (Sandhan et al., 2022d). The lack of robust morphological analyzers makes it challenging to extract morphological information, which is crucial for dependency parsing. Similarly, dependency information is essential for several downstream tasks such as word order linearisation (Krishna et al., 2019) which helps to decode possible interpretation of the poetic composition. Additionally, the ubiquitous nature of compounding in Sanskrit is difficult due to the implicitly encoded semantic relationship between its constituents (Sandhan et al., 2022c). These 4 tasks can be viewed as a preliminary requirement for developing robust NLP technology for Sanskrit. Thus, we develop novel neural-based linguistically informed architectures for all 4 tasks, reporting state-of-the-art performance on Sanskrit benchmark datasets (Sandhan et al., 2022c,d,a).

In this work, we introduce a neural Sanskrit NLP toolkit named SanskritShala² to assist computational linguistic analyses involving multiple tasks such as word segmentation, morphological tagging, dependency parsing, and compound type identification. SanskritShala is also deployed as a web

¹It means ‘a school of Sanskrit’.

²Roughly, it can be translated as ‘a school of Sanskrit’.

application that enables users to input text and gain real-time linguistic analysis from our pretrained systems. It is also equipped with user-friendly interactive data annotation capabilities that allow annotators to rectify the system when it makes errors. It provides the following benefits: (1) A user with no prior experience with deep learning can utilise it for educational purposes. (2) It can function as a semi-supervised annotation tool that requires human oversight for erroneous corrections. We publicly release the source code of the 4 modules included in the toolkit, 7 word embedding models that have been trained on publicly available Sanskrit corpora and multiple annotated datasets such as word similarity, relatedness, categorization, analogy prediction to measure the word embeddings' quality. To the best of our knowledge, this is the first neural-based Sanskrit NLP toolkit that contains a variety of NLP modules integrated with a web-based interface.

Summarily, our key contributions are as follows:

- We introduce the first neural Sanskrit NLP toolkit to facilitate automatic linguistic analyses for 4 downstream tasks (§4).
- We release 7 pretrained Sanskrit embeddings and suit of 4 intrinsic evaluation datasets to measure the word embeddings' quality (§5).
- We integrate SanskritShala with a user-friendly web-based interface which is helpful for pedagogical purposes and in developing annotated datasets (§5).
- We publicly release codebase and datasets of all the modules of SanskritShala which currently mark the state-of-the-art results.³

2 Related Work on Sanskrit NLP Tools

Recently, the Sanskrit Computational Linguistics (SCL) field has seen significant growth in building web-based tools to help understand Sanskrit texts. Goyal and Huet (2016a) introduced the Sanskrit Heritage Reader (SHR), a lexicon-driven shallow parser that aids in the selection of segmentation solutions. Samsādhani is another web-based tool consisting of various rule-based modules (Kulkarni and Sharma, 2019; Kulkarni et al., 2020; Sriram et al., 2023). Recently, Terdalkar and Bhattacharya (2021, 2022) introduced a web-based annotation

tool for knowledge-graph construction and a metrical analysis.

In short, tools for NLP can be divided into two groups: rule-based and annotation tools. Rule-based tools have limitations such as not providing a final solution, limited vocabulary coverage, and lacking user-friendly annotation features. Annotation tools, on the other hand, do not have the recommendations of rule-based systems, relying solely on annotators. To address these limitations, a web-based annotation framework called SHR++ (Krishna et al., 2020c) was proposed. It combines the strengths of both types of tools by offering all possible solutions from rule-based system SHR for tasks like word segmentation and morphological tagging, allowing annotators to choose the best solution rather than starting from scratch.

Our proposal, SanskritShala, goes a step further by integrating a neural-based NLP toolkit that combines state-of-the-art neural-based pre-trained models with rule-based suggestions through a web-based interface. Each module of SanskritShala is trained to predict the solutions from the exhaustive candidate solution space generated by rule-based systems. Hence, it makes predictions in real time using neural-based models that have already been trained. Thus, a complete solution is shown to the users / annotators, which was not possible in any of the previous attempts.

Further, annotators can easily correct the mispredictions of the system with the help of user-friendly web-based interface. This would significantly reduce the overall cognitive load of the annotators. To the best of our knowledge, SanskritShala is the first NLP toolkit available for a range of tasks with a user friendly annotation interface integrated with the neural-based modules.

3 About Sanskrit

Sanskrit is an ancient language known for its cultural significance and knowledge preservation. However, it presents challenges for deep learning due to its morphological complexity, compounding, free word order, and lack of resources. Sanskrit's intricate grammar, with its combination of roots, prefixes, and suffixes, requires advanced algorithms to analyze and understand. Compounding adds another layer of complexity as multiple words combine to form new words with unique meanings (Krishna et al., 2016; Sandhan et al., 2022c). The free word order in Sanskrit complicates tasks

³<https://github.com/Jivnesh/SanskritShala>

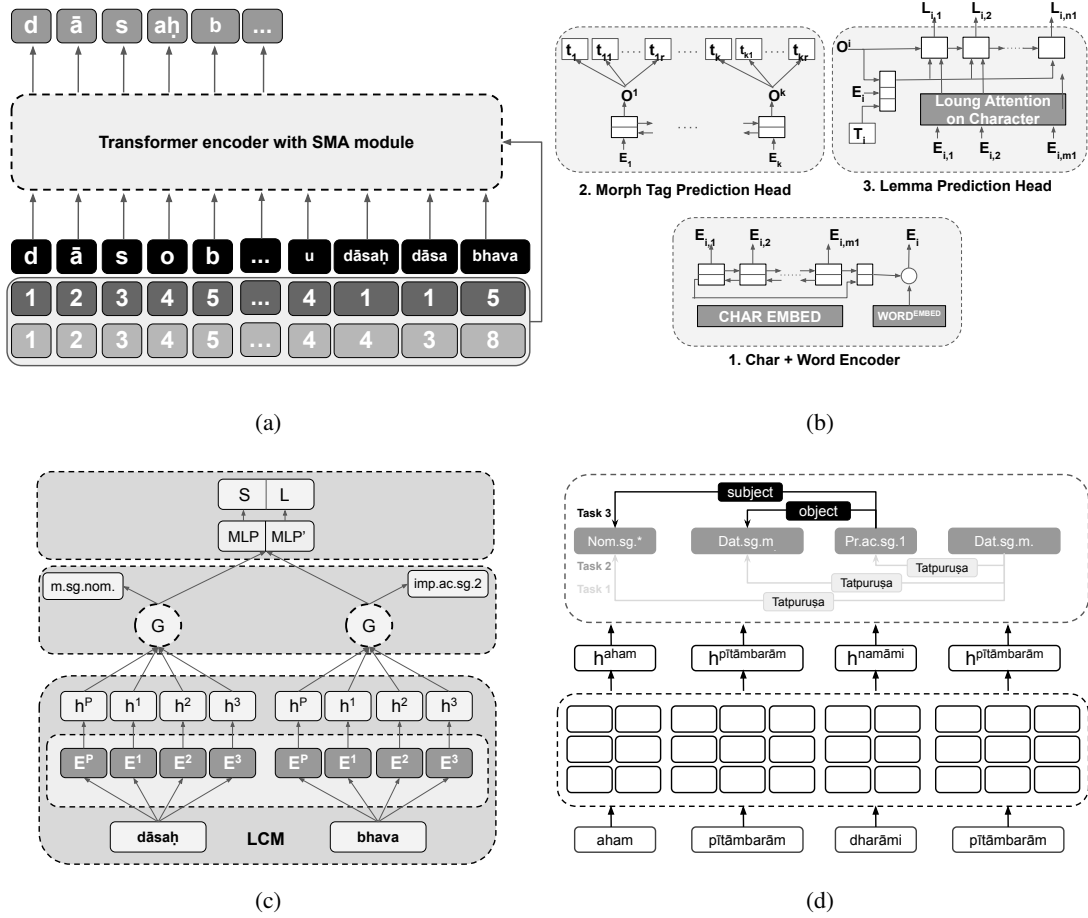


Figure 1: (a) Toy illustration of the TRANSLIST system. “dāsobhava”. Translation: “Become a servant.” (b) LemmaTag architecture in which multi-task learning formulation is leveraged to predict morphological tags and lemmas by employing bidirectional RNNs with character-level and word-level representations. (c) Proposed ensemble architecture for dependency parsing integrated with the LCM pretraining. LCM is acronym for three auxiliary tasks: Lemma prediction, Case prediction and Morphological tag prediction. (d) Toy example illustrating the context-sensitive multi-task learning system: “*aham pīta-ambaram dharāmi*” (Translation: “I wear a yellow cloth”) where ‘*pīta-ambaram*’ is a compound having *Tatpuruṣa* semantic class according to the context presented.

like parsing and understanding, requiring models to comprehend meaning regardless of word placement (Krishna et al., 2023, 2019). Moreover, Sanskrit is considered a low-resource language, lacking extensive datasets and pre-trained models (Sandhan et al., 2021). Overcoming these challenges necessitates linguistic expertise, computational techniques, and sufficient language resources. Developing specialized models to handle Sanskrit’s morphology, compounding, and word order is essential. Creating annotated datasets, lexicons, and corpora will also contribute to advancing research and applications in Sanskrit (Sandhan et al., 2022b, 2023). Despite the obstacles, utilizing deep learning to explore Sanskrit benefits the preservation of cultural heritage and facilitates a deeper understanding of India’s literature and philosophy, while also push-

ing the boundaries of natural language processing.

4 A Neural NLP Sanskrit Toolkit

In this section, we describe SanskritShala, which is a neural Sanskrit NLP toolkit designed to aid computational linguistic analysis including various tasks, such as word segmentation, morphological tagging, dependency parsing, and compound type identification. It is also available as a web application that allows users to input text and obtain real-time linguistic analysis from our pretrained algorithms. We elucidate SanskritShala by first elaborating on its key modules.

Word Tokenizer: Earlier *lexicon-driven* systems for Sanskrit word segmentation (SWS) rely on Sanskrit Heritage Reader (Goyal and Huet, 2016b,

SHR), a rule-based system, to obtain the exhaustive solution space for segmentation, followed by diverse approaches to find the most valid solution. However, these systems are rendered moot while stumbling out-of-vocabulary words. Later, *data-driven* systems for SWS are built using the most recent techniques in deep learning, but can not utilize the available candidate solution space. To overcome the drawbacks of both lines of modelling, we build a **Transformer-based Linguistically-Informed Sanskrit Tokenizer** (TransLIST) (Sandhan et al., 2022d) containing (1) a component that encodes the character-level and word-level potential candidate solutions, which tackles *sandhi* scenario typical to SWS and is compatible with partially available candidate solution space, (2) a novel soft-masked attention for prioritizing selected set of candidates and (3) a novel path ranking module to correct the mispredictions. Figure 1(a) illustrates the TransLIST architecture, where the candidate solutions obtained from SHR are used as auxiliary information. In terms of the perfect match (PM) metric, TransLIST outperforms with 93.97 PM compared to the state-of-the-art (Hellwig and Nehrdich, 2018) with 87.08 PM.

Morphological Tagger: Sanskrit is a morphologically-rich fusional Indian language with 40,000 possible labels for inflectional morphology (Krishna et al., 2020b; Gupta et al., 2020), where homonymy and syncretism are predominant (Krishna et al., 2018). We train a neural-based architecture (Kondratyuk et al., 2018, LemmaTag) on Sanskrit dataset (Krishnan et al., 2020). Figure 1(b) illustrates the system architecture in which multi-task learning formulation is leveraged to predict morphological tags and lemmas by employing bidirectional RNNs with character-level and word-level representations. Our system trained on the Sanskrit dataset stands first with 69.3 F1-score compared to the second position with 69.1 F1-score on the Hackathon dataset (Krishnan et al., 2020) leaderboard.⁴

Dependency Parser: Due to labelled data bottleneck, we focus on low-resource techniques for Sanskrit dependency parsing. Numerous strategies are tailored to improve task-specific performance in low-resource scenarios. Although these strategies are well-known to the NLP community, it is

not obvious to choose the best-performing ensemble of these methods for a low-resource language of interest, and not much effort has been given to gauging the usefulness of these methods. We investigate 5 low-resource strategies in our ensembled Sanskrit parser (Sandhan et al., 2022a): data augmentation, multi-task learning, sequential transfer learning, pretraining, cross/mono-lingual and self-training. Figure 1(c) shows our ensembled system, which supersedes with 88.67 Unlabelled Attached Score (UAS) compared to the state-of-the-art (Krishna et al., 2020a) with 87.46 UAS for Sanskrit and shows on par performance in terms of Labelled Attached Score.

Sanskrit Compound Type Identifier (SaCTI) is a multi-class classification task that identifies semantic relationships between the components of a compound. Prior methods only used the lexical information from the constituents and did not take into account the most crucial syntactic and contextual information for SaCTI. However, the SaCTI task is difficult mainly due to the implicitly encrypted context-dependent semantic relationship between the compound’s constituents. Thus, we introduce a novel multi-task learning approach (Sandhan et al., 2022c) (Figure 1(d)) which includes contextual information and enhances the complementary syntactic information employing morphological parsing and dependency parsing as two auxiliary tasks. SaCTI outperforms with 81.7 F1-score compared to the state-of-the-art by Krishna et al. (2016) with 74.0 F1-score.

5 Sanskrit Resources in SanskritShala

In this section, we describe 7 word embeddings pre-trained on Sanskrit corpora and suit of 4 intrinsic tasks datasets to assess the quality of word embeddings, followed by the description of web interface.

Pretrained word embeddings for Sanskrit:

There are two types of embedding methods: static and contextualized. Table 1 shows how they are categorized based on the smallest unit of input to the embedding model, such as character, subword, or token level. The paper focuses on two token-level word embeddings: Mikolov et al. (2013, word2vec) and Pennington et al. (2014, GloVe). Word2vec is the foundation for all subsequent embeddings and works on the local context window, while GloVe considers the global context. To address the OOV issue, subword (Wieting et al., 2016; Bojanowski

⁴Hackathon leaderboard: <https://competitions.codalab.org/competitions/35744#results>

Class	Input type	Systems
Static	character	charLM
	subword	fastText
	token	word2vec, gloVe, LCM
Contextualized	character	ELMo
	subword	ALBERT

Table 1: Overview of Sanskrit pretrained embeddings.

et al., 2017; Heinzerling and Strube, 2018) and character-level (Kim et al., 2016; Jozefowicz et al., 2016) modeling have been proposed. We also explore two contextualized embeddings: ELMo (Peters et al., 2018) and ALBERT (Lan et al., 2020), a lighter version of BERT. We trained these 6 embedding methods on Sanskrit corpora and made the pretrained models publicly available (Sandhan et al., 2023).⁵ The following section describes our proposed pretraining for low-resource settings.

LCM Pretraining: We propose a supervised pretraining, which automatically leverages morphological information using the pretrained encoders. In a nutshell, LCM integrates word representations from multiple encoders trained on three independent auxiliary tasks into the encoder of the neural dependency parser. LCM is acronym for three auxiliary tasks: Lemma prediction, Case prediction and Morphological tag prediction. LCM follows a pipeline-based approach consisting of two steps: pretraining and integration. Pretraining uses a sequence labelling paradigm and trains encoders for three independent auxiliary tasks. Later, these pretrained encoders are combined with the encoder of the neural parser via a gating mechanism similar to Sato et al. (2017). The LCM consists of three sequence labelling-based auxiliary tasks, namely, predicting the dependency label between a modifier-modified pair (**LT**), the monolithic morphological label (**MT**), and the case attribute of each nominal (**CT**). We encourage readers to refer Sandhan et al. (2021, LCM) for more details.

Datasets: The quality of word embedding spaces is evaluated through intrinsic and extrinsic methods. This study focuses on intrinsic evaluation, which involves assessing semantic and syntactic information in the words without testing on NLP applications. It is based on works such as Mikolov et al. (2013) and Baroni et al. (2014). These evaluations require a query inventory containing a query word

and a related target word. However, such query inventories are not readily available for Sanskrit. To address this, we annotated query inventories for 4 intrinsic tasks: analogy prediction, synonym detection, relatedness, and concept categorization. The inventories were constructed using resources such as Sanskrit WordNet (Kulkarni, 2017), Amarakoṣa (Nair and Kulkarni, 2010), and Sanskrit Heritage Reader (Goyal and Huet, 2016b; Huet and Goyal, 2013).

Web Interface: Figure 2 shows our Sanskrit-Shala toolkit that offers interactive web-based predictions for various NLP tasks. The toolkit is built using React framework, which makes it user-friendly and easy to use. One of the tasks it handles is the word segmentation task, which is built on top of the web-based application called SHR++. The SHR++ demonstration is depicted in Figure 3(a). The user inputs a Sanskrit string, which is then sent in real-time to SHR for potential word splits. The system prediction is then obtained from the pretrained word tokenizer. The human annotator is presented with the candidate solution space, with the system prediction highlighted in yellow. The toolkit also features a flask-based application for morphological tagging, which takes user input and scrapes possible morphological tags for each word using SHR. As shown in Figure 3(b), the predictions of the pretrained morphological tagger are displayed in green or orange, depending on whether they are present in the candidate solution of SHR or not. The user can also add a new tag if the actual tag is missing in the SHR solution space or the system’s prediction. For the dependency parsing module, we have built a react-based front-end. The user input is passed to the pretrained model to generate a dependency structure. As illustrated in Figure 4(a), the front-end automatically loads the predicted dependency tree and allows the user to make corrections if there are any mispredictions. Additionally, Figure 4(b) shows a flask-based application for the compound type identifier, where users can give input to the system through its web interface. The final annotations can be downloaded after each individual module. We plan to maintain the progress of Sanskrit NLP and offer an overview of available datasets and existing state-of-the-art via the leaderboard for various tasks.

Interactive Chatbot: SanskritShala-bot is a rule-based chatbot that makes it easy to automate simple

⁵<https://github.com/Jivnesh/SanskritShala/tree/master/EvalSan>

and repetitive user requests, like answering frequently asked questions and directing users to relevant resources. It is also easier to set up and maintain than AI-powered chatbots, which are more complicated. SanskritShala-bot is useful for addressing frequently asked standard queries. It helps familiarize users with the platform by providing them with information and guidance on how to use it. It can answer questions about the platform’s features, help users find their way around it, and explain step-by-step how to do certain tasks. This can make it easier for users to get started and leading to a better user experience.

6 Conclusion

We present the first neural-based Sanskrit NLP toolkit, SanskritShala which facilitates diverse linguistic analysis for tasks such as word segmentation, morphological tagging, dependency parsing and compound type identification. It is set up as a web-based application to make the toolkit easier to use for teaching and annotating. All the codebase, datasets and web-based applications are publicly available. We also release word embedding models trained on publicly available Sanskrit corpora and various annotated datasets for 4 intrinsic evaluation tasks to assess the intrinsic properties of word embeddings. We strongly believe that our toolkit will benefit people who are willing to work with Sanskrit and will eventually accelerate the Sanskrit NLP research.

Limitations

We plan to extend SanskritShala by integrating more downstream tasks such as Post-OCR correction, named entity recognition, verse recommendation, word order linearisation, and machine translation. Improving the performance of existing tasks would be important. For example, the current dependency parser is very fragile (performance drops by 50%) in the poetry domain.

Ethics Statement

Our work involves the development of a platform for annotating Sanskrit text. We believe that this platform will be useful for people who are willing to work with Sanskrit for research and educational purposes. We have ensured that our platform is designed ethically and responsibly. We do not foresee any harmful effects of our platform on any community. However, we caution users to use the plat-

form carefully as our pretrained models are not perfect, and errors can occur in the annotation process. All our systems are built using publicly available benchmark datasets, and we have released all our pretrained models and source codes publicly for future research. We are committed to transparency and open access in our work, and we believe that sharing our resources will benefit the wider NLP community. We also acknowledge that NLP research can have potential ethical implications, particularly in areas such as data privacy, bias and discrimination. We are committed to continuing to consider these ethical implications as we develop our platform, and we welcome feedback from the community on how we can improve our ethical practices.

Acknowledgements

We are thankful to Oliver Hellwig for the DCS dataset and Gerard Huet for the Sanskrit Heritage Engine. We are grateful to Hackathon organizers⁶ who encouraged us to build the best performing morphological tagger. We thank Amrith Krishna, Uniphore for providing the SHR++ interface as a starting point for the web interface of SanskritShala. We appreciate the assistance of Bishal Santra and Suman Chakraborty, IIT Kharagpur in deploying SanskritShala on the IIT Kharagpur server. We are grateful to Hritik Sharma, IIT Kanpur for helping us build a React-based front-end for SanskritShala. We are thankful to Hrishikesh Terdalkar, IIT Kanpur for helpful discussions on deploying systems. We appreciate the anonymous reviewers’ insightful suggestions for enhancing this work. We’d like to say thanks to everyone who helped us make the different neural models for SanskritShala. The work was supported in part by the National Language Translation Mission (NLTM): Bhashini project by Government of India. The work of the first author is supported by the TCS Fellowship under the Project TCS/EE/2011191P.

References

Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. 2021. *Automatic speech recognition in Sanskrit: A new speech corpus and modelling insights*. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5039–5050. Online. Association for Computational Linguistics.

⁶<https://sanskritpanini.github.io/index.html>

- Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. [Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. 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*, 5:135–146.
- Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. [A distributed platform for Sanskrit processing](#). In *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India. The COLING 2012 Organizing Committee.
- Pawan Goyal and Gérard Huet. 2016a. [Design and analysis of a lean interface for sanskrit corpus annotation](#). *Journal of Language Modelling*, 4:145.
- Pawan Goyal and Gérard Huet. 2016b. [Design and analysis of a lean interface for sanskrit corpus annotation](#). *Journal of Language Modelling*, 4:145.
- Ashim Gupta, Amrith Krishna, Pawan Goyal, and Oliver Hellwig. 2020. [Evaluating neural morphological taggers for Sanskrit](#). In *Proceedings of the 17th SIG-MORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 198–203, Online. Association for Computational Linguistics.
- 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. European Language Resources Association (ELRA).
- Oliver Hellwig and Sebastian Nehrlich. 2018. [Sanskrit word segmentation using character-level recurrent and convolutional neural networks](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium. Association for Computational Linguistics.
- Gérard Huet and Pawan Goyal. 2013. [Design of a lean interface for sanskrit corpus annotation](#).
- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. [Exploring the limits of language modeling](#).
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. [Character-aware neural language models](#). In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*, page 2741–2749. AAAI Press.
- Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. [LemmaTag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928, Brussels, Belgium. Association for Computational Linguistics.
- Amrith Krishna, Ashim Gupta, Deepak Garasangi, Jeevnesh Sandhan, Pavankumar Satuluri, and Pawan Goyal. 2023. [Neural approaches for data driven dependency parsing in Sanskrit](#). In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 1–20, Canberra, Australia (Online mode). Association for Computational Linguistics.
- Amrith Krishna, Ashim Gupta, Deepak Garasangi, Pavankumar Satuluri, and Pawan Goyal. 2020a. [Keep it surprisingly simple: A simple first order graph based parsing model for joint morphosyntactic parsing in Sanskrit](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4791–4797, Online. Association for Computational Linguistics.
- Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, and Pawan Goyal. 2018. [Free as in free word order: An energy based model for word segmentation and morphological tagging in Sanskrit](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2550–2561, Brussels, Belgium. Association for Computational Linguistics.
- Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2020b. [A graph-based framework for structured prediction tasks in Sanskrit](#). *Computational Linguistics*, 46(4):785–845.
- Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2021. [A Graph-Based Framework for Structured Prediction Tasks in Sanskrit](#). *Computational Linguistics*, 46(4):785–845.
- Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016. [Compound type identification in Sanskrit: What roles do the corpus and grammar play?](#) In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 1–10, Osaka, Japan. The COLING 2016 Organizing Committee.
- Amrith Krishna, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, and Pawan Goyal. 2019. [Poetry to prose conversion in Sanskrit as a linearisation task: A case for low-resource languages](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1160–1166, Florence, Italy. Association for Computational Linguistics.
- Amrith Krishna, Shiv Vidhyut, Dilpreet Chawla, Sruti Sambhavi, and Pawan Goyal. 2020c. [SHR++: An](#)

- interface for morpho-syntactic annotation of Sanskrit corpora. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7069–7076, Marseille, France. European Language Resources Association.
- Sriram Krishnan, Amba Kulkarni, and Gérard Huet. 2020. Validation and normalization of dcs corpus using sanskrit heritage tools to build a tagged gold corpus.
- Amba Kulkarni, Pavankumar Satuluri, Sanjeev Panchal, Malay Maity, and Amruta Malvade. 2020. Dependency relations for Sanskrit parsing and treebank. In *Proceedings of the 19th International Workshop on Treebanks and Linguistic Theories*, pages 135–150, Düsseldorf, Germany. Association for Computational Linguistics.
- Amba Kulkarni and Dipti Sharma. 2019. Pāṇinian syntactico-semantic relation labels. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 198–208, Paris, France. Association for Computational Linguistics.
- Malhar Kulkarni. 2017. *Sanskrit WordNet at Indian Institute of Technology (IITB) Mumbai*, pages 231–241. Springer Singapore, Singapore.
- 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*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.
- Sivaja Nair and Amba Kulkarni. 2010. The knowledge structure in amarakosa. pages 173–189.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Jivnesh Sandhan, Laxmidhar Behera, and Pawan Goyal. 2022a. Systematic investigation of strategies tailored for low-resource settings for sanskrit dependency parsing.
- Jivnesh Sandhan, Ayush Daksh, Om Adideva Paranjay, Laxmidhar Behera, and Pawan Goyal. 2022b. Prabhupadavani: A code-mixed speech translation data for 25 languages. In *Proceedings of the 6th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 24–29, Gyeongju, Republic of Korea. International Conference on Computational Linguistics.
- Jivnesh Sandhan, Ashish Gupta, Hrishikesh Terdalkar, Tushar Sandhan, Suwendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022c. A novel multi-task learning approach for context-sensitive compound type identification in Sanskrit. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4071–4083, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Jivnesh Sandhan, Amrith Krishna, Pawan Goyal, and Laxmidhar Behera. 2019. Revisiting the role of feature engineering for compound type identification in Sanskrit. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 28–44, IIT Kharagpur, India. Association for Computational Linguistics.
- Jivnesh Sandhan, Amrith Krishna, Ashim Gupta, Laxmidhar Behera, and Pawan Goyal. 2021. A little pretraining goes a long way: A case study on dependency parsing task for low-resource morphologically rich languages. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 111–120, Online. Association for Computational Linguistics.
- Jivnesh Sandhan, Om Adideva Paranjay, Komal Digmurthi, Laxmidhar Behera, and Pawan Goyal. 2023. Evaluating neural word embeddings for Sanskrit. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 21–37, Canberra, Australia (Online mode). Association for Computational Linguistics.
- Jivnesh Sandhan, Rathin Singha, Narein Rao, Suwendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022d. Translist: A transformer-based linguistically informed sanskrit tokenizer.
- Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain Universal Dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada. Association for Computational Linguistics.
- Krishnan Sriram, Amba Kulkarni, and Gérard Huet. 2023. Validation and normalization of DCS corpus and development of the Sanskrit heritage engine’s segmenter. In *Proceedings of the Computational*

Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference, pages 38–58, Canberra, Australia (Online mode). Association for Computational Linguistics.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2021. [Sangrahaka: A tool for annotating and querying knowledge graphs](#). In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2021*, page 1520–1524, New York, NY, USA. Association for Computing Machinery.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2022. [Chandojnanam: A sanskrit meter identification and utilization system](#).

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. [Charagram: Embedding words and sentences via character n-grams](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas. Association for Computational Linguistics.

LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models

Victor Dibia

Microsoft Research

victordibia@microsoft.com

Abstract

Systems that support users in the automatic creation of visualizations must address several subtasks - understand the semantics of data, enumerate relevant visualization goals and generate visualization specifications. In this work, we pose visualization generation as a multi-stage generation problem and argue that well-orchestrated pipelines based on large language models (LLMs) and image generation models (IGMs) are suitable to addressing these tasks. We present LIDA, a novel tool for generating grammar-agnostic visualizations and infographics. LIDA comprises of 4 modules - A SUMMARIZER that converts data into a rich but compact natural language summary, a GOAL EXPLORER that enumerates visualization goals given the data, a VISGENERATOR that generates, refines, executes and filters visualization code and an INFOGRAPHER module that yields data-faithful stylized graphics using IGMs. LIDA provides a python api, and a *hybrid* USER INTERFACE (direct manipulation and *multilingual* natural language) for interactive chart, infographics and data story generation. Code and demo are available at this url - <https://microsoft.github.io/lida/>

1 Introduction

Visualizations make data accessible by reducing the cognitive burden associated with extracting insights from large tabular datasets. However, visualization authoring is a complex creative task, involving multiple steps. First the user must build familiarity with the dataset (content and semantics) and enumerate a set of relevant goals or hypotheses that can be addressed using the data. Next, users must select the right visualization representation (marks, transformations and layout) for each goal. Finally, the user must implement the visualization either as code or using available direct manipulation interfaces. Each of these steps require expertise, and can be tedious as well as error

prone for *users with limited visualization experience* (novices). Existing research has sought to address these challenges by *automating* the visualization (AUTOVIZ) creation process, given a dataset (Podo et al., 2023). *Automation* may occur in two modes: i.) fully automated - the system automatically generates visualizations relevant to the data ii.) semi-automated - the user specifies their goals and the system generates visualizations that address these goals. The former mode is valuable for users unfamiliar with the data and the latter is valuable for users with some familiarity with the data and the visualization task.

Consequently, a successful AUTOVIZ tool must excel at each of several *subtasks* - understand the semantics of the data, enumerate relevant visualization goals and generate visualization specifications that meet syntax, design, task and perceptual requirements of these goals (Podo et al., 2023). Furthermore, given the target demographic (novice users), such a tool must support the user by offering NL (NL) interaction modalities (Mitra et al., 2022; Narechania et al., 2020; Chen et al., 2022), affordances to control system behavior and sense making tools to understand and debug/verify system behavior. While related work has addressed aspects of the AUTOVIZ task, there are several known limitations (Podo et al., 2023) such as they: (i) rely on heuristics that are limited in coverage, challenging to craft and tedious to maintain (Wongsuphasawat et al., 2017). (ii) require significant user interaction to generate visualizations (Wongsuphasawat et al., 2017; Moritz et al., 2018). (iii) implement automated approaches that offer limited control over system input and output (Dibia and Demiralp, 2019) (iv) require grammar (or chart type) specific training data and model architectures (Dibia and Demiralp, 2019; Luo et al., 2018) for each sub task, (v) do not consider alternative chart representation formats such as infographics.

Concurrently, advances in large foundation mod-

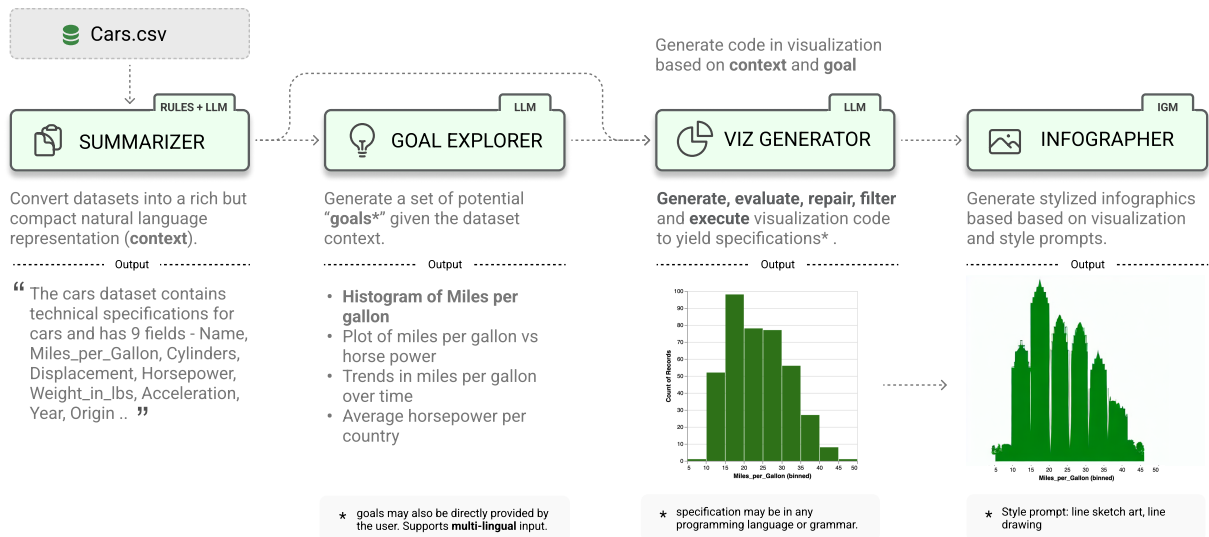


Figure 1: LIDA generates visualizations and infographics across 4 modules - data summarization, goal exploration, visualization generation and infographics generations. Example output from each module is shown.

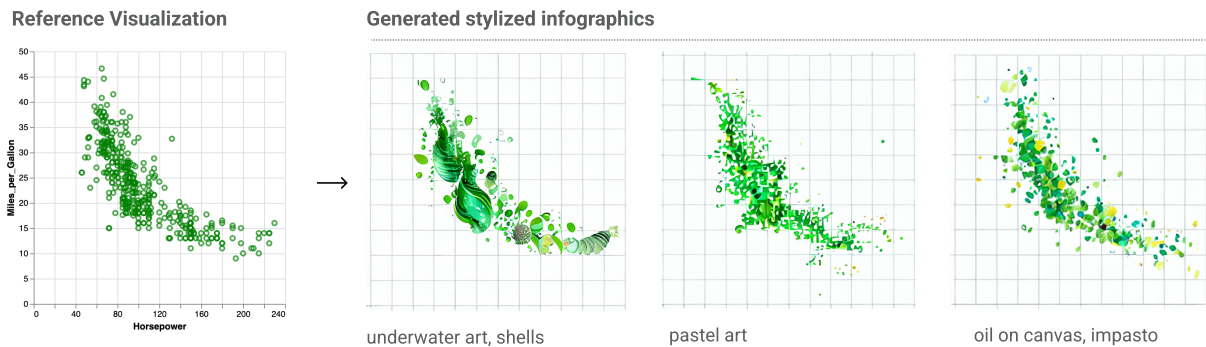


Figure 2: Example *data-faithful* infographics and associated style prompts generated with LIDA.

els (Bommasani et al., 2021) have shown state of the art performance on a variety of creative tasks such as multilingual text generation, code generation, image captioning, image generation, and image editing. In this work, we argue that the vast capabilities of these models can be *assembled* to address the AUTOVIZ task, whilst *addressing the limitations of existing approaches*. This work makes the following contributions:

- We present a novel multi-stage, modular approach (Fig 1) for the automatic generation of data visualization and infographics using LLMs¹. Specifically, we (i) Efficiently represent datasets as NL summaries, suitable as grounding context for an LLM to address visualization tasks. (ii) Generate a set of visualization goals using LLMs. Importantly, we leverage prompt engineering to steer the model towards generat-

¹This work primarily utilizes the OpenAI *gpt-3.5-turbo-x* line of models for text and code generation.

ing *correct* visualization that follow *best practices* (see Appendix C). (iii) Apply LLMs to generate grammar-agnostic visualization specification based on generated (or human provided) goals. (iv) Provide a *hybrid interface* that supports traditional direct manipulation controls (e.g., manually select which fields to explore) and a rich **multilingual** NL interface to support user’s with varied skill/experience. (v) Apply text-conditioned image generation models (IGM) models in generating stylized infographics that are both informative (generally faithful to data), aesthetically pleasing, memorable and engaging (see section 2.3).

- We introduce metrics for evaluating LLM-enabled visualization tools, including a metric for pipeline reliability (visualization error rate - VER), and visualization quality (self-evaluated visualization quality - SEVQ) (see section 4).

- We implement our approach in an Open Source library - LIDA². LIDA provides a python api, a web api and a **rich web interface** useful for research and practical applications.

Compared to existing AUTOVIZ approaches, LIDA proposes an implementation that is **simplified** (eliminates the need for subtask-specific models), **general** (can be adapted to generate visualizations in any programming language or grammar), **flexible** (individual modules can be optimized) and **scalable** (the system performance will *improve* with advances in the underlying LLM). Taken together, these contributions provide building blocks towards complex workflows such as visualization translation, *chart question answering* (with applications in accessibility of charts), automated *data exploration* and *automated data stories*.

To the best of our knowledge, LIDA is the first tool to formulate visualization/infographic generation as a multi-step generation task and demonstrate an end-to-end pipeline that addresses a variety of subtasks.

2 Related Work

LIDA is informed by research on large foundation models applied to creative tasks across modalities such as text and images, and advances in automated generation of visualizations and infographics.

2.1 Foundation Models for Creative Tasks

Advances in large transformer-based (Vaswani et al., 2017) models trained on massive amounts of data (terabytes of text and images) have led to a paradigm shift where a single model demonstrates state of the art task performance across multiple data modalities such as text, images, audio and video. These models, also known as foundation models (Bommasani et al., 2021), have been shown to be effective for a variety of *human creativity* tasks. LLMs like the GPT3 series (Brown et al., 2020), OPT (Zhang et al., 2022), PALM (Chowdhery et al., 2022), LAMBDA (Cohen et al., 2022) learn complex semantics of language allowing them to be effective in tasks such as text summarization, question answering. Code LLMs such as Codex (Chen et al., 2021), AlphaCode (Li et al., 2022), InCoder (Fried et al., 2022) show state of the art performance on a suite of code intelligence tasks. Finally, models such as CLIP (Radford et al.,

2021), DALLE (Ramesh et al., 2022, 2021) and Latent Diffusion (Rombach et al., 2022) have shown state of the art capabilities on image generation tasks such as image captioning, image editing, and image generation.

In this work, we adopt insights from Program-Aided Language models (Gao et al., 2022) - a setup where LLMs generate programs as the intermediate reasoning steps, but offload the solution step to a runtime such as a python interpreter. We leverage the *language modeling* capabilities of LLMs in generating *semantically meaningful* visualization goals, and their *code writing* capabilities in generating *visualization code* which is compiled to yield visualizations. These visualizations (images) are then used as input to image generation models in generating stylized infographics.

2.2 Automated Visualization (AUTOVIZ)

Extant AUTOVIZ research have explored multiple approaches such as heuristics, task decomposition or learning based approaches. Heuristics-based approaches explore properties of data in generating a search space of potential visualizations (Wongsuphasawat et al., 2017), ranking these visualizations based on quality attributes (Luo et al., 2018; Moritz et al., 2018) and presenting them to the user. For example, DeepEye (Luo et al., 2018) enumerates all possible visualizations and classifies/ranks them as “good” or “bad” using a binary decision tree classifier while Voyager (Wongsuphasawat et al., 2017) uses heuristics to enumerate the space of visualizations. However, heuristics can be tedious to maintain, may have poor coverage of the visualization space and does not leverage information encoded in existing datasets. More recent work has explored a task decomposition approach where the AUTOVIZ process is decomposed into multiple tasks that are solved individually via specialized tools and aggregated to yield visualizations (Narechania et al., 2020; Chen et al., 2022; Wang et al., 2022b). For example NL4DV (Narechania et al., 2020) implements a custom query engine that parses natural language queries, identifies attributes/tasks and generates Vega-Lite specifications. A limitation of task decomposition approaches is that they are bottlenecked by the implementation performance for each step (e.g., limitations with models for disambiguating natural language queries as seen in NL4DV (Narechania et al., 2020)). Finally, end-to-end learning-based

²<https://microsoft.github.io/lida/>.

approaches seek to automatically learn mappings from data directly to generated visualizations. For example, Data2Vis (Dibia and Demiralp, 2019) (the most relevant work to this study) uses a sequence to sequence model that implicitly addresses AUTOVIZ subtasks by learning a mapping from raw JSON data sampled from datasets to Vega-Lite (Satyanarayan et al., 2017) specifications. Some limitations of current learning approaches is that they are limited to a single grammar, require custom models, custom paired training data and training objectives (Dibia and Demiralp, 2019; Luo et al., 2018; Chen et al., 2022) for each supported grammar, and do not provide a path to generating infographics. Furthermore, they do not provide mechanisms for fine-grained control of visualization output or provide robust error detection and recovery strategies.

LIDA addresses these limitations in several ways: (i) Leverages patterns learned by LLMs from massive language and code dataset, applying this knowledge to subtasks. (ii) Provides a single grammar-agnostic pipeline that generates visualization in multiple programming languages and visualization grammars. (iii) Supports natural language based control of generated visualizations. (iv) leverage emergent capabilities of large language models such chain of thought reasoning to improve reliability of generated text/code (Kojima et al., 2022; Wei et al., 2022; Shi et al., 2022a), model calibration (Kadavath et al., 2022) (predictions on correctness probabilities of visualizations) as well as self-consistency (Wang et al., 2022a) in ranking/filtering results. (v) provides a mechanism for generating infographics that are data-faithful and aesthetically pleasing. (vi) supports a fully automatic mode where an LLM is used to discover meaningful goals/hypotheses (fields to visualize, questions to ask) or a semi automatic mode where the user provides a hypothesis and it generates a visualization.

By choosing to cast visualization/infographic generation as generation tasks that offloads core problem solving to LLMs and IGMs, LIDA simplifies the design and maintenance of such systems.

2.3 Infographics Generation

Infographics (information graphics) are visual artifacts that seek to convey complex data-driven narratives using visual imagery and embellishments (Harrison et al., 2015). Existing research has shown that infographics are aesthetically pleasing, engag-

ing and more memorable (Tyagi et al., 2021; Harrison et al., 2015; Haroz et al., 2015), at no additional cost to the user (Haroz et al., 2015). These properties have driven their applications in domains like fashion, advertisement, business and general communications. However, the creation of infographics that convey data insights can be a tedious process for content creators, often requiring skills across multiple tools and domains. Research on infographic generation have mainly explored the creation of pictographs (Haroz et al., 2015) - replacing the marks on traditional charts with generated images and learning to extract/transfer styles from existing pictographs (Shi et al., 2022b). In this work, we extend this domain to exploring the generation of both visual marks as well as generating the entire infographic based on natural language style descriptions using large image generation models such as DALLE (Ramesh et al., 2022, 2021) and Latent Diffusion (Rombach et al., 2022). This approach also enables user-generated visual styles and personalization of visualizations to fit user preferences such as color palettes, visual styles, fonts etc.

3 The LIDA System

LIDA comprises of 4 core modules - a SUMMARIZER, a GOAL EXPLORER, a VISGENERATOR and an INFOGRAPHER (see Fig 1). Each module is implemented in the LIDA [github repo](#) as a python library with an optional user interface (see Appendix A).

3.1 SUMMARIZER

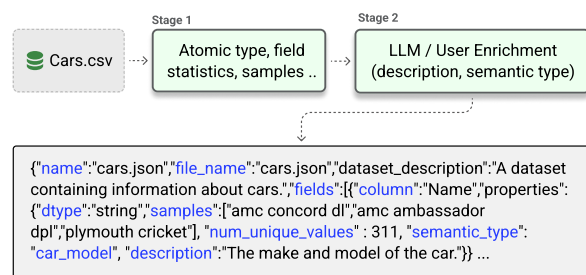


Figure 3: The SUMMARIZER module constructs a NL summary from extracted data properties (atomic types, field statistics) and an optional LLM enrichment (predicted field descriptions, semantic types).

LLMs are capable zero shot predictors, able to solve multiple tasks with little or no guiding examples. However, they can suffer from hallucination e.g., generating text that is not grounded in training data

or the current task. One way to address this is to *augment* (Mialon et al., 2023) the LLM with grounding context. Thus, the goal of the summarizer is to produce an **information dense** but **compact**³ summary for a given dataset that is *useful as grounding context* for visualization tasks. A useful context is defined as one that *contains information an analyst would need to understand the dataset and the tasks that can be performed on it*. The summary is implemented in two stages (see Fig 3)

Stage 1 - Base summary generation: We apply rules in extracting dataset properties including atomic types (e.g., integer, string, boolean) using the pandas library (McKinney, 2010), general statistics (min, max, # unique values) and a random non-null list of n samples for each column.

Stage 2 - Summary enrichment: The base summary is optionally enriched by an LLM or a user via the LIDA ui to include semantic description of the dataset (e.g., a dataset on the technical specification of cars), and fields (e.g., miles per gallon for each car) as well as field semantic type prediction (Zhang et al., 2019).

3.2 GOAL EXPLORER

This module generates data exploration goals, given a summary generated by the SUMMARIZER. We express goal generation as a multitask generation problem where the LLM must generate a *question* (hypothesis), a *visualization* that addresses the question and *rationale* (see Fig 4). We find that requiring the LLM to produce a rationale leads to more semantically meaningful goals.

```
{ "question": "What is the distribution of Miles_per_Gallon?",
  "visualization": "Histogram of Miles_per_Gallon",
  "rationale": "This tells us about the fuel efficiency of the cars in the dataset and how it is distributed." }
```

Figure 4: A goal generated by LIDA is a JSON data structure that contains a question, a visualization and a rationale.

3.2.1 VISGENERATOR

The VISGENERATOR generates visualization specifications and is comprised of 3 submodules - a *code scaffold constructor*, a *code generator* and a *code executor*.

Code scaffold constructor: Implements a library of code scaffolds that correspond to programming

³Note: the summary must be compact in order to maximize the limited context token budget of LLMs.

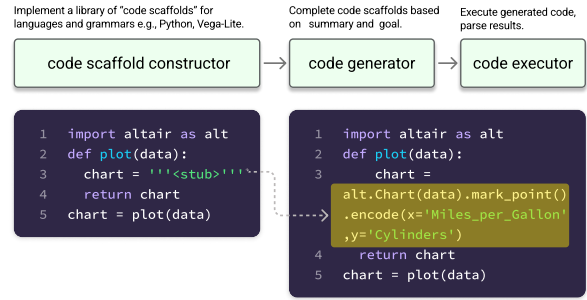


Figure 5: The VISGENERATOR module constructs visualization code scaffolds, fills a constrained section (< stub >) and executes the scaffold.

languages and visualization grammars e.g., python scaffolds support grammars such as Matplotlib, GGPlot, Plotly, Altair, Seaborn, and Bokeh. Each scaffold is an *executable program* that i.) imports relevant dependencies ii.) defines an empty function stub which returns a visualization specification (see Fig 5a).

Code generator: Takes a scaffold, a dataset summary, a visualization goal, and builds a prompt. An LLM (applied in *fill-in-the-middle* mode (Bavarian et al., 2022)) is then used to generate n candidate visualization code specifications.

Code executor: Post-processes and executes⁴ the code specifications as well as filters the results. LIDA implements several filtering mechanisms to detect errors, each with latency tradeoffs: (i) generates a large sample for n with high temperature, discard candidates that do not compile. (ii) apply self consistency (Wang et al., 2022a) in LLMs where multiple candidates are generated and the solution with the highest consensus is selected. (iii) generate correctness probabilities (Kadavath et al., 2022) for all candidates and selects the one with the highest probability. Note that the last two approaches are computationally expensive (require multiple forward passes through an LLM) and are not suitable for real time applications. The final output is a list of visualization specifications (code) and associated raster images.

3.2.2 VIZOPS - Operations on Generated Visualizations

Given that LIDA represents visualizations as code, the VISGENERATOR also implements submodules to perform operations on this representation.

Natural language based visualization refinement: Provides a conversational api to iteratively

⁴Execution in a sandbox environment is recommended.

refine generated code (e.g., *translate chart t hindi ... zoom in by 50%* etc) which can then be executed to generate new visualizations.

Visualization explanations and accessibility: Generates natural language explanations (valuable for debugging and sensemaking) as well as accessibility descriptions (valuable for supporting users with visual impairments).

Visualization code self-evaluation and repair: Applies an LLM to *self-evaluate* generated code on multiple dimensions (see section 4.1.2).

Visualization recommendation: Given some context (goals, or an existing visualization), recommend additional visualizations to the user (e.g., for comparison, or to provide additional perspectives).

3.3 INFOGRAPHER

This module is tasked with generating stylized graphics based on output from the VISGENERATOR module (see Fig 2). It implements a library of visual styles described in NL that are applied directly to visualization images. Note that the style library is editable by the user. These styles are applied in generating infographics using the text-conditioned image-to-image generation capabilities of diffusion models (Rombach et al., 2022), implemented using the *Peacasso* library api (Dibia, 2022). An optional post processing step is then applied to improve the resulting image (e.g., replace axis with correct values from visualization, removing grid lines, and sharpening edges).

3.4 USER INTERFACE

LIDA implements a user interface that communicates with the core modules over a REST and Websocket api. The user interface implements several views.

Data upload and summarization: This view allows the user to upload a dataset and explore a sample of rows in the dataset via a table view. A data upload event triggers a call to the SUMMARIZER and GOAL EXPLORER module and displays a summary of the dataset and a list of potential goals. This view also allows the user to optionally annotate and refine the generated summary or curate fields used in the dataset.

Visualization view: This view allows the user to optionally provide a visualization goal in NL (e.g., "what is the fuel efficiency per country?") or select a generated goal and then displays a generated visualization. For each visualization, intermediate output from the models (underlying data sum-

mary, visualization specification, code scaffold) are shown as explanations to aid in sensemaking, and debugging (see Fig 9). This view also implements the VIZOPS capabilities described in Section 3.2.2 (e.g., See the interface for visualization evaluation in Fig 10). Note that the NL interface inherits the multilingual language capabilities of the underlying LLM, enabling multilingual NL interaction.

Overall, the combination of these modules result in a system that is able to implicitly address an array of data visualization operations such as data transformation, encoding, mark selection, styling, layout, and annotation (Wang et al., 2022b).

4 Evaluation

4.1 Evaluation Metrics

Our initial evaluation of LIDA focuses on two high level metrics - visualization error rates (VER) to provide signals on the *reliability* of the LIDA pipeline, and self-evaluated visualization quality (SEVQ) to assess the *quality* of generated visualizations.

4.1.1 Visualization Error Rate (VER)

Visualization error rate is computed as the percentage of generated visualizations that result in code compilation errors. This metric provides critical insights into the reliability of the LIDA pipeline and impact of changes to the system (e.g., prompt engineering or scaffold update).

$$VER = \frac{E}{T} * 100$$

Where: - E = Number of generated visualizations with code compilation errors, and - T = Total number of generated visualizations.

4.1.2 Self-Evaluated Visualization Quality (SEVQ)

Recent work shows LLMs like GPT-4 encode broad world knowledge (OpenAI, 2023), can assess the quality of their output (Kadavath et al., 2022; Lin et al., 2022) and can approximate human judgments for tasks such as summarization (Liu et al., 2023). Our observations applying GPT3.5/GPT-4 to visualization tasks suggest similar results. Specifically, GPT-4 has learned to encode *some* visualization best practices and can apply these in generating critiques of visualization code across multiple dimensions. Thus, to evaluate visualization quality, we compute an SEVQ metric by applying GPT-4 in assessing the quality of generated visualizations. Specifically, we task GPT-4

with scoring generated visualization code (a numeric value from 1-10 and a rationale) across 6 dimensions - code accuracy, data transformation, goal compliance, visualization type, data encoding, and aesthetics. These dimensions are *informed* by existing literature on visualization generation/recommendation e.g., Wang et al. (2022b) outline 6 visualization tasks including data transformation, encoding, marks, styling, layout and annotation, while (Moritz et al., 2018) codify constraints for visualization quality across expressivity (does it convey the facts of the data) and effectiveness (is the information more readily perceived compared to other visualizations) criteria. Additional details on prompts used for each dimension are provided in Appendix B.

4.2 Evaluation Benchmark Settings

Our initial benchmark is based on 57 datasets sourced from the vega datasets repository⁵. For each dataset, LIDA is tasked with generating 5 goals and 1 visualization per goal across multiple grammars⁶. For reproducibility, we set *temperature* = 0 and number of samples *n* = 1 for the LLM. A gallery of the generated evaluation visualizations can be viewed on the LIDA [project page](#).

4.3 Evaluation and Ablation Study Results

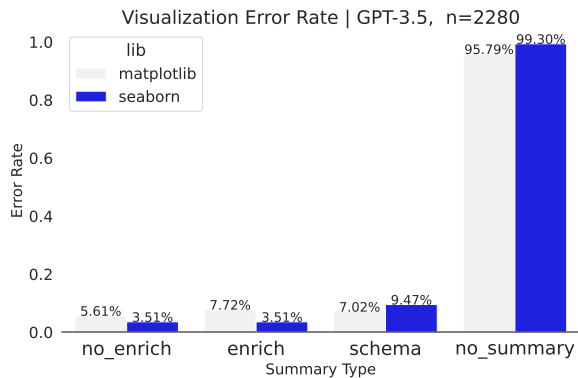


Figure 6: Results from an ablation study on the impact of data summarization strategies on visualization error rate (VER) metric.

Overall, we find that LIDA is able to generate visualizations with a low error rate (VER = 3.5%). We also conduct an ablation study to inform on the impact of the SUMMARIZER across the fol-

⁵<https://github.com/vega/vega-datasets>

⁶LIDA is given a single try for each step. In theory, the error rates can be driven to zero, by recursively applying the visualization self-evaluation and self-repair modules.

lowing conditions - (i) *no_enrich*: a base summary with no enrichment (see Section 3.1), (ii) *enrich*: summary with LLM enrichment, (iii) *schema*: only field names, i.e., schema as summary, and (iv) *no_summary*: no summary. Results show that including a summary leads to reduced error rate compared to simply adding field names (schema) as summary. We also find that enriching the base summary with an LLM has less of an effect on VER (with variations across visualization grammar), and an expressive, well-represented grammar like Seaborn having lower VER. These results are summarized in Figure 6. We also find that the SEVQ metric is valuable in identifying semantic quality issues with generated visualizations. For example, Fig 10 shows an example where the user has requested a pie chart, and the LIDA self-evaluation module critiques this visualization using the SEVQ metric, providing a rationale for why a bar chart is more effective (see Fig 10), with the option to automatically repair the visualization.

5 Conclusion

In this work, we formulate visualization generation as a multi-stage text (and code) generation problem that can be addressed using large language models. We present LIDA - a tool for the automatic generation of grammar-agnostic visualizations and infographics. LIDA addresses limitations of current automatic visualization systems - automatic generation of hypothesis/goals given datasets, conversational interface for controllable visualization generation and refinement, support for multiple visualization grammars using the same pipeline and the ability to generate infographics. LIDA is effective compared to state of the art systems (see example [gallery](#) of generated visualizations); it offers a simplified system implementation and leverages the immense language modeling and code generation capabilities of LLMs in implicitly solving complex visualization subtasks. Finally, we introduce metrics for assessing reliability (visualization error rate - VER) and visualization quality (self-evaluated visualization quality -SEVQ) for LLM-enabled visualization tools. We hope modules implemented in LIDA will serve as useful building blocks in enabling complex creative workflows such as *visualization translation*, *chart question answering*(with applications in accessibility of charts), *automated data exploration* and *automated storytelling*.

6 Limitations

While LIDA demonstrates clear advances in how we can support users in authoring visualizations and infographics, there are several limitations that offer a natural avenue for future research.

Low Resource Grammars: The problem formulation introduced in LIDA depends on the underlying LLMs having *some* knowledge of visualization grammars as represented in *text and code* in its training dataset (e.g., examples of Altair, Vega, Vega-Lite, GGPlot, Matplotlib, *represented in* Github, Stackoverflow, etc.). For visualization grammars not well represented in these datasets (e.g., tools like Tableau, PowerBI, etc., that have graphical user interfaces as opposed to code representations), the performance of LIDA may be limited without additional model fine-tuning or translation. Furthermore, performance may be limited for complex tasks (e.g., tasks requiring complex data transformations) beyond the expressive capabilities of specific grammars. Further research is needed to: i.) study effects of strategies like task disambiguation ii.) impact of task complexity and choice of programming language/grammar on performance.

Deployment and Latency: Large language models (e.g., GPT3.5 used in this work) are computationally expensive and require significant compute resources to deploy at low latency. These costs can prove to be impractical for *real-world application*. In addition, the current setup includes a code execution step which is valuable for verification but increases deployment complexity (requires a sandbox). Thus, there is opportunity to: i.) train smaller capable LLMs (Touvron et al., 2023) finetuned on a curated dataset of programming languages and visualization grammars ii.) design vulnerability mitigation approaches such as limiting program scope or generating only input parameters for visualization grammar compilers.

Explaining System Behavior: The approach discussed in this paper simplifies the design of visualization authoring systems, but also inherits interpretability challenges associated with large language models. While LIDA offers intermediate outputs of the model (e.g., generated code and specifications) as *explanations*, as well as post-hoc explanations of generated code (see section 3.2.2), there is a need for further research in explaining system behavior (conditions when they are needed) and providing actionable feedback to the user.

System Evaluation: Benchmarking LLM’s on cre-

ativity tasks can be challenging. While the current study introduces metrics for evaluating reliability (VER) and visualization quality (SEVQ) (see section 4), there is a need for more comprehensive benchmarks on a variety of datasets and visualization grammars. Furthermore, there are research opportunities to i.) study and quantify the capabilities of LLMs in *encoding* and *applying* visualization best practices ii.) conduct empirical studies that evaluate model behavior, mapping out failure cases and proposing mitigations iii.) qualitatively study the impact of tools like LIDA on user *creativity* while authoring visualizations.

Acknowledgements

This manuscript has benefited from comments and discussions with members of the HAX group (Saleema Amershi, Adam Fourny, Gagan Bansal), VIDA group (Steven Drucker, Dan Marshall), Bongshing Lee, Rick Barraza and others at Microsoft Research.

References

- Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*.
- Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.
- 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.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Qiaochu Chen, Shankara Pailoor, Celeste Barnaby, Abby Criswell, Chenglong Wang, Greg Durrett, and Işıl Dillig. 2022. Type-directed synthesis of visualizations from natural language queries. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA2):532–559.

- 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*.
- Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung ching Chang, Claire Cui, Cosmo Du, Daniel De Freitas Adiwardana, Dehao Chen, Dmitry (Dima) Lepikhin, Ed H. Chi, Erin Hoffman-John, Heng-Tze Cheng, Hongrae Lee, Igor Krivokon, James Qin, Jamie Hall, Joe Fenton, Johnny Soraker, Kathy Meier-Hellstern, Kristen Olson, Lora Moïs Aroyo, Maarten Paul Bosma, Marc Joseph Pickett, Marcelo Amorim Menegali, Marian Croak, Mark Díaz, Matthew Lamm, Maxim Krikun, Meredith Ringel Morris, Noam Shazeer, Quoc V. Le, Rachel Bernstein, Ravi Rajakumar, Ray Kurzweil, Romal Thoppilan, Steven Zheng, Taylor Bos, Toju Duke, Tulsee Doshi, Vincent Y. Zhao, Vinodkumar Prabhakaran, Will Rusch, YaGuang Li, Yanping Huang, Yanqi Zhou, Yuanzhong Xu, and Zhifeng Chen. 2022. Lamda: Language models for dialog applications. In *arXiv*.
- Victor Dibia. 2022. [Interaction design for systems that integrate image generation models: A case study with peacasso](#).
- Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46.
- Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. InCoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.
- Steve Haroz, Robert Kosara, and Steven L Franconeri. 2015. Isotype visualization: Working memory, performance, and engagement with pictographs. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 1191–1200.
- Lane Harrison, Katharina Reinecke, and Remco Chang. 2015. Infographic aesthetics: Designing for the first impression. In *Proceedings of the 33rd Annual ACM conference on human factors in computing systems*, pages 1187–1190.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.
- Yang Liu, Dan Iter, Yichong Xu, Shuhang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.
- Yuyu Luo, Xuedi Qin, Nan Tang, Guoliang Li, and Xinran Wang. 2018. Deepeye: Creating good data visualizations by keyword search. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD*, pages 1733–1736.
- Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.
- Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.
- Rishab Mitra, Arpit Narechania, Alex Endert, and John Stasko. 2022. Facilitating conversational interaction in natural language interfaces for visualization. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pages 6–10. IEEE.
- Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. 2018. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448.
- Arpit Narechania, Arjun Srinivasan, and John Stasko. 2020. N14dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.
- OpenAI. 2023. [Gpt-4 technical report](#).
- Luca Podo, Bardh Prenkaj, and Paola Velardi. 2023. Machine learning for visualization recommendation systems: Open challenges and future directions. *arXiv preprint arXiv:2302.00569*.

- 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.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.
- Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685.
- Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-lite: A grammar of interactive graphics. *IEEE TVCG (Proc. InfoVis)*.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022a. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.
- Yang Shi, Pei Liu, Siji Chen, Mengdi Sun, and Nan Cao. 2022b. Supporting expressive and faithful pictorial visualization design with visual style transfer. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):236–246.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#).
- Anjul Tyagi, Jian Zhao, Pushkar Patel, Swasti Khurana, and Klaus Mueller. 2021. User-centric semi-automated infographics authoring and recommendation. *arXiv preprint arXiv:2108.11914*.
- 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.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Yun Wang, Zhitao Hou, Leixian Shen, Tongshuang Wu, Jiaqi Wang, He Huang, Haidong Zhang, and Dongmei Zhang. 2022b. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *ACM CHI*.
- Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2019. Sato: Contextual semantic type detection in tables. *arXiv preprint arXiv:1911.06311*.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A The LIDA Library

LIDA is implemented as a python library with modules for each of the components described in Section 3. The library is available on github⁷ and can be installed using pip - `pip install lida`. The library provides a python api, web api for integration into other applications, and a command line interface. It also provides a web-based user interface for users to interact with LIDA (Fig 10, 9).

```

1 # pip install lida
2
3 from lida.modules import Manager
4
5 lida = Manager()
6 summary = lida.summarize("data/cars.csv")
7 goals = lida.generate_goals(summary, n=1)
8
9 vis_specs = manager.generate_viz(summary=summary,
10 goal=goals[i])
11 charts = manager.execute_viz(code_specs=vis_specs,
12 data=manager.data, summary=summary)
13 print(charts)

```

Figure 7: Example usage of LIDA shows how to generate a summary, visualization goals, code specifications and execute the code to generate visualizations.

B Self-Evaluated Visualization Quality (SEVQ) Prompts

For the SEVQ metric, we use GPT-4 to assess visualization quality by scoring generated visualization

⁷<https://github.com/microsoft/lida>

Select a visualization library/grammar

- Seaborn
- Altair
- Matplotlib
- Seaborn
- GGPlot

Generation Settings

Model: gpt-4-0314, n: 1, number of Temperature: 0...

Click or drag file to this area to upload
Upload .json or .csv files to generate a visualization.

Don't have data? Try any of the files below

stocks.csv cars.json wheat.json movies.json seattle-weather.csv sp500.csv

Data Summary

An enriched representation of the data (with predicted semantic types and descriptions)

seattle-weather.csv This dataset contains weather data for Seattle from 2012 to 2015.

data summary | seattle-weather.csv

Date	# Precipitation	# Temp_max	# Temp_min	# Wind	Weather
date date The date of the weather observation. # Unique values: 1461 min: 2012-01-01, max: 2015-12-31	number precipitation The amount of precipitation in inches. # Unique values: 111 0, max: 55.9 std: 6.680	number temperature The maximum temperature in degrees Fahrenheit. # Unique values: 67 min: -1.6, max: 35.6 std: 7.350	number temperature The minimum temperature in degrees Fahrenheit. # Unique values: 55 min: -7.1, max: 18.3 std: 5.023	number wind_speed The wind speed in miles per hour. # Unique values: 79 min: 0.4, max: 9.5 std: 1.438	category weather_condition The weather condition observed. # Unique values: 5

View raw summary?

Figure 8: In the data upload section of the LIDA UI, users can select a grammar of choice and upload a dataset. A dataset upload event triggers a goal generation as well as visualization generation tasks.

Visualization Generation

Select a goal above or describe a new visualization goal to generate a visualization.

What is the distribution of weather conditions observed as a pie chart

Generate

What is the distribution of weather conditions observed as a pie chart

What is the distribution of weather conditions observed as a pie chart

Weather Condition	Percentage
rain	43.87%
sun	43.81%
fog	6.91%
drizzle	3.63%
snow	1.78%

How was this visualization created? See the specifications and code below.

```

Python Code
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

def plot(data: pd.DataFrame):
    weather_counts = data['weather'].value_counts()
    plt.pie(weather_counts, labels=weather_counts.index, autopct='%1.2f%%')
    plt.title("What is the distribution of weather conditions observed as a pie chart", wrap=True)
    plt.legend(weather_counts.index, loc="best")
    return plt;

```

Refine Explain Evaluate Recommend!

Figure 9: The visualization generation section of the LIDA UI enables the user to i.) specify their overall goal in natural language and generate visualizations ii.) inspect, edit and execute generated code iii.) view the generated visualization. iv.) perform operations on generated code e.g., refine, explain, evaluate and recommend visualizations.

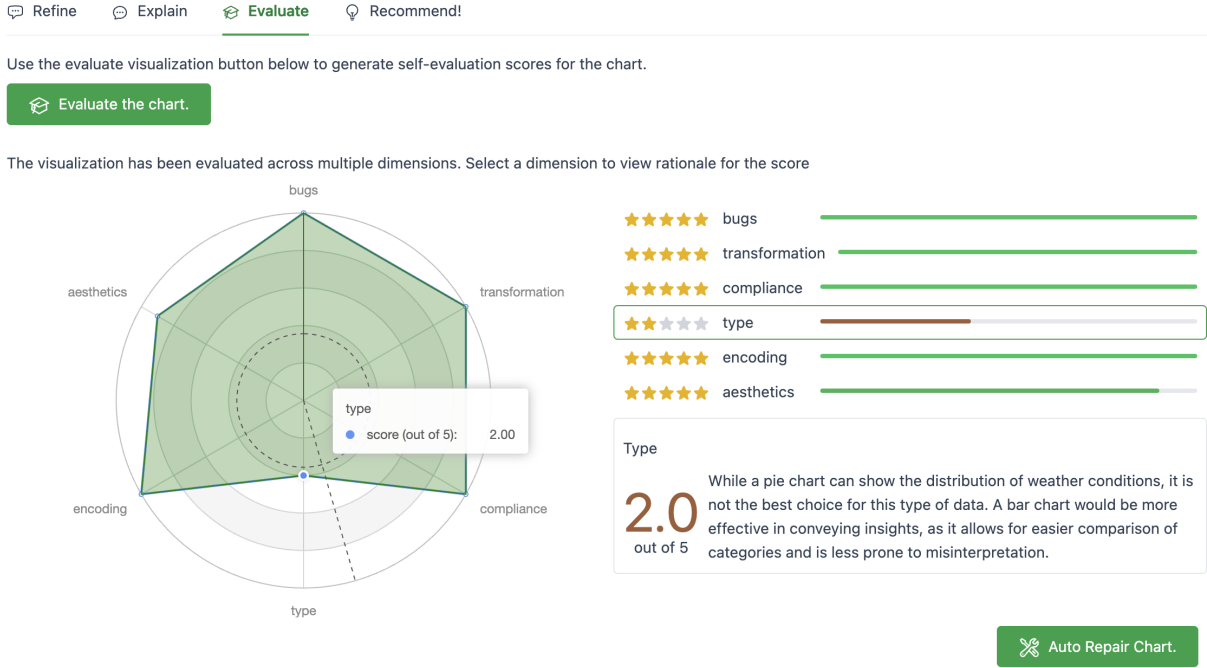


Figure 10: The self-evaluation module in LIDA is used to evaluate/critique a generated visualization, providing scores across 6 dimensions with rationale. In this case, the visualization contains a pie chart, and a bar chart is recommended as an alternative.

code across the 6 task dimensions - code accuracy, data transformation, goal compliance, visualization type, data encoding, and aesthetics. These dimensions are implemented as prompts to an LLM⁸, which then generates a score between 1-10 for each dimension. The final SEVQ score is the average of the 6 scores. A sketch of the prompts used for each dimension are enumerated in table 1.

C Design Reflections

Building a system that leverages foundation models (text and images) involves engineering decisions across a wide design space. In this section, we briefly reflect on some of the design choices we made for LIDA components and the tradeoffs we considered.

C.1 Prompt Engineering

We explored multiple approaches to building prompts that maximized the probability of the LLM solving each subtask.

- **SUMMARIZER:** We found that improving the richness of the summary (qualitative NL description, including semantic types) was *critical* to improved quality of generated goals and

⁸Exact prompts can be found at the project repository <https://github.com/microsoft/lida>.

Dimension	Prompt
Code accuracy	Does the code contain bugs, logic errors, syntax error or typos? How serious are the bugs? How should it be fixed?
Data transformation	Is the data transformed appropriately for the visualization type?
Goal compliance	How well the code meets the specified visualization goals?
Visualization type	Considering best practices, is the visualization type appropriate for the data and intent? Is there a visualization type that would be more effective in conveying insights?
Data encoding	Is the data encoded appropriately for the visualization type?
Aesthetics	Are the aesthetics of the visualization appropriate and effective for the visualization type and the data?

Table 1: Summary of the evaluation dimensions and the corresponding prompt sketches.

visualization code. Implementation wise, we began with a manually crafted summary of the data (see Section 3.1), and then enriched it via calls to an LLM *and* optional user refinement of the summary.

- **GOAL EXPLORER:** Providing few shot examples in the prompts where fields and rationale

are linked via symbols (e.g., plot a histogram of field X vs Y to show relationship between X and Y) nudges the model to use exact dataset field names, and minimizes the occurrence of hallucinated fields. Prompt engineering also provides mechanisms to bake in visualization best practices e.g. *avoid pie charts, apply visualization best practices, Imagine you are a highly experienced visualization specialist and data analyst.*

- **VISGENERATOR:** Casting visualization code generation as a *fill-in-the-middle* problem (as opposed to free-from completion) ensures the model to generate executable code *focused* on the task. For example, in Fig 5, the model is *instructed* to generate only the `< stub >` portion of the code scaffold. We also note that the degrees of freedom allotted to the model (e.g., specifying how much of the scaffold to complete) can influence its ability to add tasks with varied complexity. For example, a scaffold that allows the model generate data preprocessing code (and includes libraries like statsmodels etc) allows the model to address tasks that require steps such as data transformation, sampling and statistical analysis before generating visualizations etc.
- Overall, we found that setting a low temperature ($t = 0$; generating the most likely visualization) coupled with a per-grammar code scaffold provided the best results in terms of yielding code that correctly compiles into visualization specifications and faithfully addresses the subtask. We also explored prompt formulations that addressed multiple tasks to minimize costs (latency and compute). For example, summary enrichment is a single call where the LLM must generate dataset descriptions, field descriptions and semantic types.

C.2 Infographic Generation

We found that setting a low *strength* parameter ($0.25 < strength < 0.45$) for the latent diffusion model (image-to-image mode) and using parsimonious style prompts resulted in stylized images that were faithful to the general *structure* of the original visualization, minimizing distorted or irrelevant imagery. This sort of controlled generation is *necessary* to avoid the distraction (Haroz et al., 2015) that can arise from superfluous imagery in infographics.

C.3 Natural Language Interaction

(i) **HYBRID INTERFACE:** Providing a hybrid interface that allows traditional direct manipulation steps in creating visualizations (e.g., selecting which fields to use), paired with a NL interface allows users to leverage existing mental models with traditional visualization tools as well as the NL affordances of LIDA. (ii) **NL INTERACTION MODES:** Beyond generating a base visualization, we also enable operations on generated visualization code (e.g., refinement, explanation, evaluation, recommendation). This builds on insights from Mitra et al. (2022) who propose multi-turn dialog interfaces for visualization authoring towards resolving ambiguities.

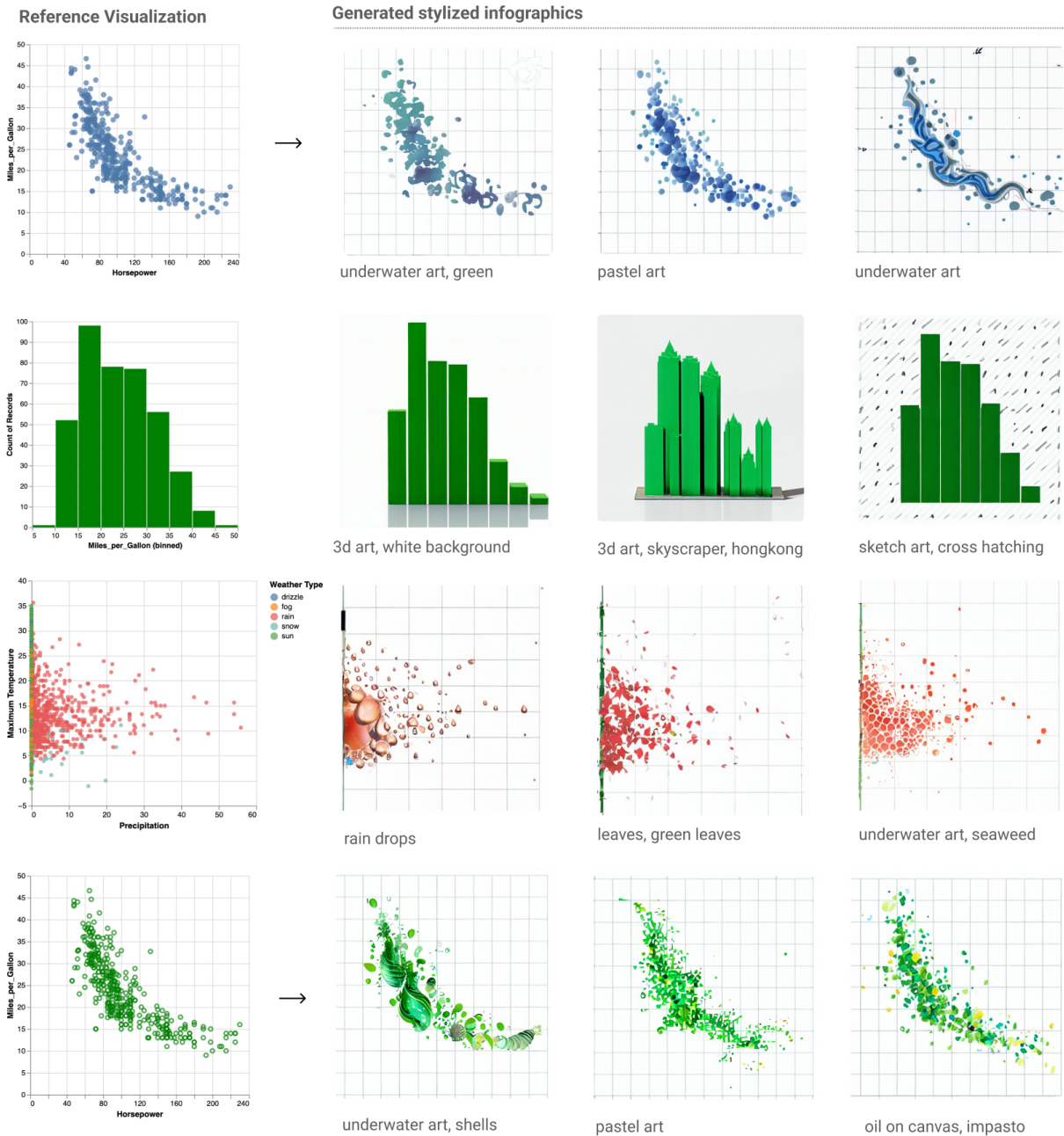


Figure 11: The LIDA infographer module supports the generation of data-faithful infographics. Each infographic is conditioned on a generated visualization as well as natural language style tags which can be used to customize the appearance of the chart.

MetaPro Online: A Computational Metaphor Processing Online System

Rui Mao^{*♦}, Xiao Li[♦], Kai He[♦], Mengshi Ge[♦] and Erik Cambria[♦]

[♦]Nanyang Technological University, Singapore

[♦]Ruimao Tech, Shenzhen, China

{mao.r, li.x, he.kai}@ruimao.tech; {mengshi.ge, cambria}@ntu.edu.sg

Abstract

Metaphoric expressions are a special linguistic phenomenon, frequently appearing in everyday language. Metaphors do not take their literal meanings in contexts, which may cause obstacles for language learners to understand them. Metaphoric expressions also reflect the cognition of humans via concept mappings, attracting great attention from cognitive science and psychology communities. Thus, we aim to develop a computational metaphor processing online system, termed MetaPro Online¹, that allows users without a coding background, e.g., language learners and linguists, to easily query metaphoricity labels, metaphor paraphrases, and concept mappings for non-domain-specific text. The outputs of MetaPro can be directly used by language learners and natural language processing downstream tasks because MetaPro is an end-to-end system.

1 Introduction

Metaphors are frequently used in our everyday language. Relevant statistics (Cameron, 2003; Martin, 2006; Steen et al., 2010) found that about one-third of sentences in corpora contain metaphors. It is possible to use approximately six metaphors during a minute of speaking (Geary, 2011).

Metaphors are a non-negligible linguistic phenomenon for linguistic learners and researchers. On the other hand, Lakoff and Johnson (1980) argued that metaphors are not just a linguistic phenomenon, because metaphors also reflect human cognition. Human perceptions of LOVE are not complete without metaphors, e.g., love is a *journey*, *magic*, *attraction*, *madness*, *nurturance*², etc.

There are two important research domains in computational metaphor processing: linguistic and conceptual. The former area focuses on the surface

* The work was done when he worked at Ruimao Tech.

¹Website: <https://metapro.ruimao.tech/>

²Italics denotes metaphors.

AllenNLP

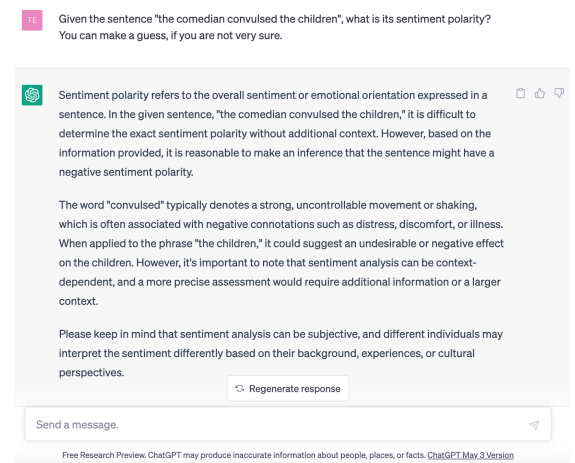
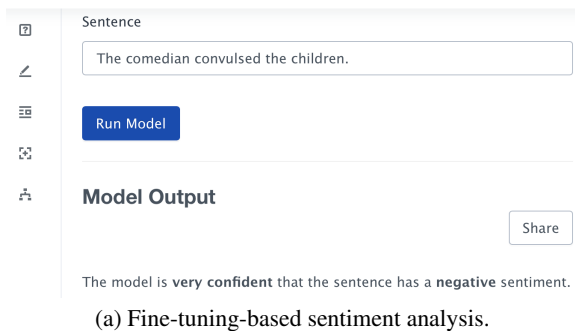


Figure 1: Errors in current AI applications, caused by a metaphoric expression.

realization of metaphors, e.g., metaphor identification and interpretation.

The motivation is that metaphors do not take their literal meanings in contexts, which may cause difficulties for language learners and machines to

MetaPro 2.0: A Computational Metaphor Processing System

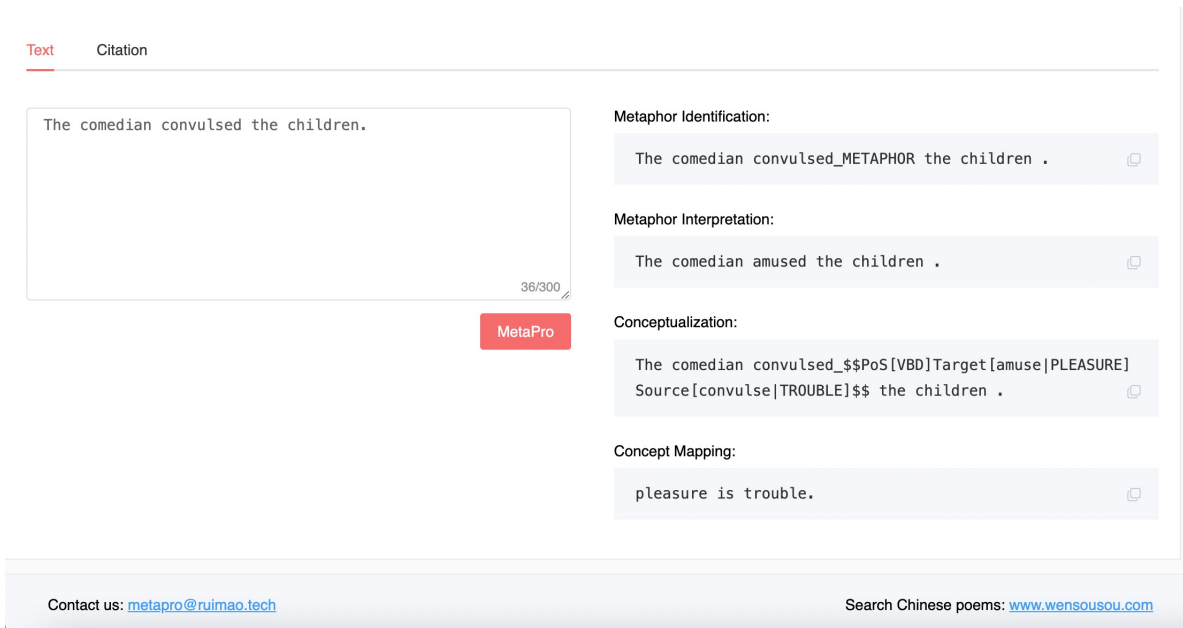


Figure 2: MetaPro Online.

understand the real meanings of metaphors (Cambria et al., 2017). As seen in Figure 1 (a), given the metaphoric expression, “the comedian *convulsed* the children”, a sentiment classifier yields an incorrect negative label for the sentence. In Figure 1 (b), the generative AI seems to explain why such an error occurs – “The word ‘convulsed’ typically denotes a strong, uncontrollable movement or shaking, which is often associated with negative connotations”. AI notes that “convulsed” triggers the negative prediction for the sentence based on its literal meaning. Figure 1 (c) demonstrates that literally translating a metaphor into another language may cause interpretation difficulties because of cultural differences. For example, the given sentence is translated as “the comedian caused the children convulsions” in Chinese, which carries a very different meaning from the real English meaning. Thus, a linguistic metaphor processing system (Mao et al., 2022) aims to automatically identify metaphors and paraphrase them into their literal counterparts to mitigate the interpretation barrier for users.

Conceptual metaphor processing focuses on identifying and mapping source and target concept domains of metaphors. The motivation is to understand the cognition of humans from their metaphoric expressions. For example, given a metaphoric expression, “she *attacked* his argu-

ment”, Lakoff and Johnson (1980) argued that conceptually the ARGUMENT concept (target domain) is mapped to a WAR concept (source domain). One uses WAR to metaphorically describe ARGUMENT, reflecting strategies, attack, and defense behaviors associated with ARGUMENT. Lakoff and Johnson (1980) also believed that these concepts naturally frame our thinking and behaviors when arguing. Researchers can study cognition from metaphor concept mappings (see Section 6). Thus, a conceptual metaphor processing system aims to automatically generate source and target concept agents for a metaphor to represent the concept mappings.

Many previous works focused on metaphor identification tasks (Mao et al., 2019; Feng and Ma, 2022), metaphor interpretation tasks (Shutova, 2010; Mao et al., 2018), and concept mappings (Rosen, 2018; Ge et al., 2022). However, these methods cannot process metaphors from end to end. We proposed an end-to-end metaphor identification and interpretation method, the first version of MetaPro (Mao et al., 2022), where metaphoric single words can be paraphrased into their literal counterparts, and metaphoric multi-word expressions can be explained with clauses. However, a programming package cannot be easily used by non-programmers, e.g., language learners and linguistic researchers. On the other hand, automatic concept mappings have not been achieved on the

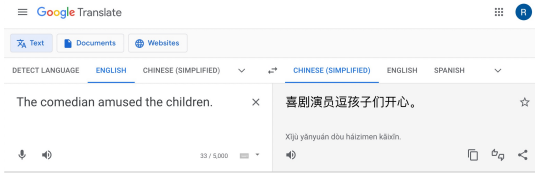
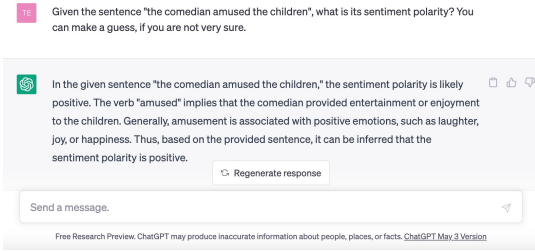
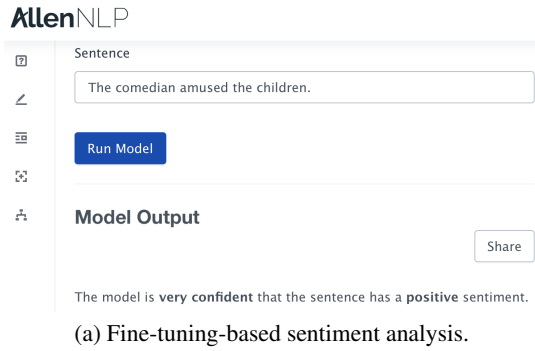


Figure 3: Correct outputs after MetaPro pre-processing.

sentence level, although we proposed a concept mapping method on the word-pair level in the work of Ge et al. (2022). Thus, we aim to develop a computational metaphor processing online system, MetaPro Online, allowing users without a coding background to easily query metaphoricity labels, metaphor interpretations, and concept mappings for non-domain-specific text.

As seen in Figure 2, given a query sentence, “the comedian *convulsed* the children”, MetaPro can identify “*convulsed*” as a metaphor. Then, the sentence is paraphrased as “the comedian amused the children” for interpretation. The source concept is TROUBLE. The target concept is PLEASURE. Then, the concept mapping is represented as PLEASURE IS TROUBLE. We can also observe that the sentiment analysis classifier, the generative AI, and the machine translation system can all yield satisfying outputs after using the paraphrased sentence in Figure 3. The concept mapping “PLEASURE IS TROUBLE”, generated by MetaPro, implies that extreme happiness can be accompanied by trouble, e.g., causing the children convulsions. Such

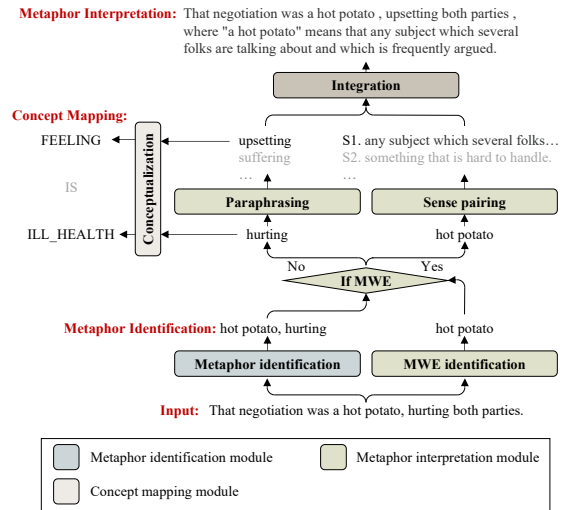


Figure 4: The framework of MetaPro. MWE denotes multi-word expressions.

implicit meanings can be used for analyzing human cognition because metaphors are stimulated by subjective experiences (Grady, 1997).

2 Related Work

According to the survey of Ge et al. (2023), there have been metaphor identification (Mao et al., 2019; Feng and Ma, 2022), interpretation (Shutova, 2010; Mao et al., 2018), and concept mapping (Rosen, 2018; Ge et al., 2022) research works in the computational linguistics community, while readily used end-to-end systems for supporting downstream applications are rare. Martin (1990) presented a metaphor interpretation system named Metaphor Interpretation, Denotation, and Acquisition System (MIDAS). The system was developed to answer metaphorical queries about Unix. Narayanan (1997) proposed a Knowledge-based Action Representations for Metaphor and Aspect (KARMA) system. The system is based on hand-coded executing schemes for motion-verb concept reasoning in an economic domain. Barn- den and Lee (2002) proposed a first-order logic rule-based system for concept mapping analysis in the mental state description domain. However, the limited manual rules in the aforementioned works cannot be used for analyzing text from a broader domain. Mao et al. (2022) proposed a non-domain-specific end-to-end system for metaphor identification and interpretation. They evaluated the system on a sentiment analysis task. However, it cannot generate concept mappings and does not provide a graphical user interface for non-programmers.

3 MetaPro

The user interface of MetaPro Online is succinct. There is just an input window and a processing button (see Figure 2). The back-end consists of three modules: metaphor identification, metaphor interpretation, and concept mappings. The overall framework can be viewed in Figure 4.

First, the metaphor identification module detects metaphors on the token level, given an input sentence, e.g., “that negotiation was a *hot potato*, *hurting* both parties”. Next, the metaphor interpretation module paraphrases the identified metaphoric single-word expressions (e.g., “*hurting*” means “upsetting”). It explains metaphoric multi-word expressions with clauses (e.g., “where ‘a hot potato’ means that any subject which several folks are talking about and which is frequently argued”). Finally, the concept mapping module abstracts the source (“ILL_HEALTH”) and target (“FEELING”) concepts of a metaphor via the lemma of the original metaphoric word (“hurt”) and the lemma of its paraphrase (“upset”), respectively. The mapping is represented as “a target concept is a source concept” (e.g., FEELING IS ILL_HEALTH). The current version of MetaPro Online cannot paraphrase and conceptualize metaphoric multi-word expressions. Thus, we explain them via clauses and omit their concept mapping generations.

We introduce each module below at the concept level to help readers understand the mechanisms of MetaPro Online. Since MetaPro is an ensemble system combining three research outcomes, we omit the algorithmic details here. One may refer to our previous works, Mao and Li (2021); Mao et al. (2022); Ge et al. (2022), to understand the technical details of metaphor identification, metaphor interpretation, and concept mapping modules, respectively.

3.1 Metaphor identification

The used algorithm of our metaphor identification module was first proposed in the work of Mao and Li (2021). We embed this algorithm into MetaPro Online because it achieved state-of-the-art performance on token-level metaphor identification tasks.

Metaphor identification is a binary classification task, classifying a word as metaphoric or literal. Our metaphor identification module uses a multi-task learning framework, simultaneously learning sequential metaphor identification and Part-of-Speech (PoS) tagging tasks. The motivation is that

previous works (Wu et al., 2018; Su et al., 2020) found PoS tags are effective features for learning metaphor identification. Thus, we introduced the learning of PoS tagging as an auxiliary task to fuse the features learned from PoS tagging.

To boost the multi-task learning framework, we also proposed a novel soft-parameter sharing mechanism, Gated Bridging Mechanism (GBM) in the work of Mao and Li (2021). The intuition is that GBM allows useful information from a neighbor tower to pass through the gate and fuse with hidden states learned in the private tower, while the gates filter out the useless information. We demonstrated the effectiveness of the proposed multi-task learning model on both metaphor identification and aspect-based sentiment analysis tasks.

3.2 Metaphor interpretation

The algorithm for the metaphor interpretation module of MetaPro was first proposed in the work of Mao et al. (2022). We embed this algorithm into MetaPro because it can paraphrase metaphoric single-word expressions and explain metaphoric multi-word expressions. The paraphrases and explanations help users understand the intended meanings of metaphors and can be processed by other downstream natural language processing systems because the outputs of the interpretation module are also natural language.

Given an identified metaphor, if the metaphor is a single-word expression, RoBERTa-large (Liu et al., 2019) and WordNet (Fellbaum, 1998) are used for paraphrasing the metaphor. We masked out a metaphor token and used the masked word prediction of RoBERTa to predict the probability distribution of candidate words. The candidate words are sourced from the WordNet hypernyms and synonyms of the lemma of the metaphor, providing paraphrase constraints for a metaphor. The hypothesis is that a metaphor can be paraphrased into one of its hypernyms and synonyms (Mao et al., 2018). The word forms of the candidate words are aligned with the original metaphor. In the work of Mao et al. (2022), we developed a word form alignment dictionary, e.g., { . . . , ‘upset’ : { ‘VBG’: ‘upsetting’, ‘VBD’: ‘upset’, . . . }, . . . } by parsing a Wikipedia dump. Thus, a lemma can map to any form, given a Penn Treebank PoS label (Marcus et al., 1993).

Mao et al. (2022) also developed a dictionary- and rule-based algorithm to identify metaphoric

multi-word expressions. They did not use a neural network model, because the identified metaphoric multi-word expressions are finally mapped to their dictionary explanations for metaphor interpretation. The dictionary- and rule-based algorithm can directly identify and map them to their dictionary explanations without using another model for mapping. In order to improve the generalization ability of this algorithm, Mao et al. (2022) developed two feature sets, namely a lemma feature set that consists of the lemmas of multi-word expressions and a triplet feature set that consists of dependency triplet features. Both feature sets can map features with the corresponding multi-word expressions. An input sentence is pre-processed with lemmatization and dependency parsing first. If features from any feature sets are the subsets of the pre-processed sentence, the corresponding multi-word expressions are detected. If there is an overlap between an identified multi-word expression and metaphoric tokens given by the metaphor identification module, the multi-word expression is explained via a clause by using its selective dictionary explanation. The final metaphor interpretation output integrates the paraphrases of single-word metaphors and metaphoric multi-word expressions. The word form of a paraphrase was aligned with the original metaphor during the candidate word preparation stage.

3.3 Concept mapping

The used algorithm of our concept mapping module was first proposed in the work of Ge et al. (2022). We embed this algorithm into MetaPro, because it can abstract concepts for words. The effectiveness of the abstracted concepts has been proved in metaphor identification and human evaluation.

The conceptual metaphor theory was proposed by Lakoff and Johnson (1980). They empirically summarized some examples of concept mappings to explain their theory. However, there is no theoretical research to explain how to conceptualize metaphors and generate mappings. For example, it was not clearly defined, if a target concept should be abstracted from the paraphrase or the context word of a metaphor. By viewing concept mapping examples given by Lakoff and Johnson (1980); Lakoff (1994), evidence from both sides can be observed. Thus, we choose to generate target concepts from the paraphrases of metaphors. The source concepts are given by the original metaphor.

Another challenge is there is no theoretical re-

search about what the abstractness level of a concept agent should be to represent a word, to our best knowledge. Thus, we follow the hypothesis of Ge et al. (2022) that an appropriate concept agent should represent the main senses of a word.

We developed a conceptualization algorithm based on WordNet and a statistical knee point algorithm (Satopaa et al., 2011) in the work of Ge et al. (2022). First, a word, e.g., a metaphor or a paraphrase, was aligned to its nominal form via WordNet and a gestalt pattern matching algorithm (Ratcliff and Metzner, 1988). Next, given a noun, we retrieved all hypernym paths from the noun node to the root node. Different paths represent different senses of the noun. The hypernyms on different levels of a path denote concepts with different abstractness levels because WordNet is a conceptually structured knowledge base. Next, the hypernyms on each path are rated by a linear score function. A higher score denotes that the hypernym is more abstract. The overall score of a hypernym is given by the sum of the hypernym scores on all its distributed paths. Next, we computed the knee point with the overall scores of all hypernyms. A hypernym is selected as a concept agent if it covers the same number of senses as the knee point hypernym and it is more concrete (a lower score) than the knee point hypernym. Otherwise, the knee point hypernym is selected as the concept agent.

Based on the above method, we can compute a source concept with a metaphor, and a target concept with a paraphrase. Then, the concept mapping is given by “a target concept is a source concept”.

3.4 Training data and lexical resources

We use VU Amsterdam Metaphor Corpus (Steen et al., 2010) as the data source to train the metaphor identification module. The training set combines the training and validation sets prepared by Leong et al. (2018), including nominal, verbal, adjective, and adverb metaphors from conversations, fiction, academic text, and news. For the statistics of the dataset, please view the work of Mao et al. (2022).

Metaphor paraphrases are based on WordNet and masked word predictions. Thus, no training set is required for learning metaphor paraphrases. The metaphoric multi-word expression interpretation is a dictionary- and rule-based algorithm. The feature and explanation dictionaries contain idiomatic multi-word expressions that were sourced

from The Idioms³ and the collection of Agrawal et al. (2018). We have defined 3,560 lemma pairing features and 3,470 dependency triplet pairing features for 3,050 idiomatic multi-word expressions in the work of Mao et al. (2022). On average, each multi-word expression has 2.7 explanations.

The concept mapping module is based on a statistical learning algorithm, using WordNet as the only lexical resource. Thus, we do not use any training set to learn concept mappings.

4 Supporting Downstream Tasks

MetaPro was used as a text pre-processing technique, where the metaphor interpretation outputs were fed into sentiment analysis classifiers instead of the original inputs. We observe that MetaPro improved the performance of Vader (Hutto and Gilbert, 2014), AllenNLP sentiment analysis⁴, and Azure sentiment analysis⁵ on a financial news headline sentiment analysis task (Cortis et al., 2017) by 1.5%, 2.2%, and 4.7% accuracy, respectively, compared with the results given by the original news headlines (Mao et al., 2022). On the other hand, we also observe that using MetaPro-generated concept mappings as features could bring a classifier extra gains in accuracy (+1.1% and +1.9%) for a depression detection task (Han et al., 2022). The benchmarking dataset was from the work of Shen et al. (2017). More importantly, the concept mapping features help the classifier explain the common concept mappings between depressed and non-depressed groups. Besides the accuracy gains, this is particularly helpful for cognitive science because MetaPro provides an automatic solution for analyzing concept mapping patterns via metaphors at scale.

Detailed performance benchmarking of different technical components of MetaPro with state-of-the-art baselines on diverse evaluation tasks and datasets can be viewed from the works of Mao and Li (2021); Mao et al. (2022); Ge et al. (2022).

5 Evaluation

Besides the performance improvements and explainability enhancements in downstream tasks, we also qualitatively evaluate the practicality of MetaPro according to the criteria proposed by Shutova (2015) in the following sections. She

³<https://www.theidioms.com/>

⁴<https://allennlp.org/>

⁵<https://azure.microsoft.com/>

proposed to evaluate a computational metaphor processing system from two aspects, namely the levels of analysis and applicability.

5.1 Levels of analysis

Linguistic metaphor. MetaPro has the capacity to analyze various forms of linguistic metaphors, encompassing both conventional and novel metaphors, including single- and multi-word expressions. It possesses the capability to handle a diverse range of linguistic metaphors without restriction on specific syntactic constructions, thus offering a comprehensive approach to metaphor processing.

Conceptual metaphor. MetaPro can abstract source and target concepts from original single-word metaphors and paraphrases, respectively. However, the current version is incapable of conceptualizing metaphoric multi-word expressions and metaphoric sentences.

Extended metaphor. The current version of MetaPro cannot process extended metaphors on the document level, due to the training set of the metaphor identification module does not contain extended metaphors. We cannot find a helpful dataset to study extended metaphors to our best knowledge.

Metaphorical inference. The current version of MetaPro cannot process metaphorical inference on the document level, because we cannot find a helpful dataset to study metaphorical inference to our best knowledge.

5.2 Applicability

Task coverage. MetaPro can identify metaphors and interpret them from both linguistic and conceptual perspectives.

Easily to integrate. The outputs of metaphor interpretation and concept mapping modules are natural language. Thus, MetaPro can be integrated with other natural language processing systems.

Unrestricted text. MetaPro can process unrestricted real-world natural language text. The current version cannot directly process emojis and spelling errors commonly appearing on social media. The maximum input length is 512 tokens after Byte-Pair Encoding (Radford et al., 2019). For efficient computing, we set up the maximum input length as 300 characters for MetaPro Online at the current version.

Be open-domain. MetaPro is not domain-specific. The training data for metaphor identification were sourced from VU Amsterdam Metaphor Corpus,

including diverse topics, different genres, and concept domains. Our metaphor interpretation is based on a pre-trained language model, trained with open-domain text without fine-tuning. The used knowledge bases, e.g., WordNet and multi-word expression dictionaries, offer general semantic knowledge of English words.

Task and knowledge dependency. Current metaphor interpretation and concept mapping modules of MetaPro depend on WordNet and multi-word expression processing dictionaries. This is because we cannot find a large annotated dataset to train neural network models to achieve the functions from end to end in a supervised fashion. We use WordNet for supporting the task of paraphrasing metaphors because simply using a medium size pre-trained language model can hardly yield satisfying results in the context of unsupervised learning.

Word class and syntax diversity. The inputs of MetaPro are sentences with diverse syntactic constructions. The current version of MetaPro targets to identify, interpret and conceptualize open-class metaphors. It also explains metaphoric multi-word expressions that contain other PoS. We focus on open-class metaphors because they contain richer semantic information than closed-class ones. This is more helpful for downstream tasks.

6 Use Case

Besides the examples in Figures 1 and 3, we reported the use cases of MetaPro in sentiment analysis in the work of Mao et al. (2022). For example, given “Rio Tinto CEO Sam Walsh rejects fears over China growth, demand”, the three examined classifiers yielded incorrect “negative” predictions. This is probably because “fears” and “reject” likely appear in negative contexts. However, after MetaPro paraphrasing the original input as “Rio Tinto CEO Sam Walsh eliminates concerns over China growth, demand”, the classifiers can yield correct “positive” predictions.

Han et al. (2022) reported the use cases in depression detection, where concept mappings are additional features besides the original text. They believe that metaphor concept mappings reflect the inner world of people because they were not explicitly presented in the text. For example, they found that LEVEL IS IMPORTANCE is a representative concept mapping for depressed people. This may result in more stress, if a person frequently maps an objective measure in the LEVEL concept

to a subjective feeling concept, e.g., IMPORTANCE.

7 Conclusion

We proposed MetaPro online in this work, which is a computational metaphor processing online system. Compared with previous works, MetaPro can identify metaphors, paraphrase them into their literal counterparts, and generate concept mappings from end to end. The system can process unrestricted and non-domain-specific English text. The user interface is very friendly to non-programmers. Thus, it can help language learners to understand the real meanings of English metaphors. We also demonstrated the performance improvements of using MetaPro on downstream AI applications, e.g., using MetaPro to automatically obtain concept mappings from social media posts to study cognitive patterns exhibited by individuals diagnosed with depression. The above use cases show that MetaPro has huge application potential in diverse domains.

However, the current version cannot paraphrase and conceptualize metaphoric multi-word expressions, which is important for sentic computing (Cambria et al., 2022). It cannot process non-English text, extended metaphors, and metaphorical inference as well. We will fill this gap in future work and strive to enhance the precision and information processing capacities of MetaPro by developing more advanced algorithms, thereby providing enhanced support for linguistic and cognitive science research endeavors.

Ethics and Broader Impact Statement

This article follows the ACL Code of Ethics. We comply with the licenses of all used datasets. Although there was no sensitive data used for training our models or developing our knowledge bases in MetaPro, we encourage all downstream applications can honor the ethical code for conducting linguistic and cognitive research. The broader impacts include but are not limited to using the tool to study the cognitive patterns of a certain group of people or a person, and using this tool to falsify original text. According to Mao et al. (2023), there are certain biases in pre-trained language models. We cannot guarantee that MetaPro can yield unbiased outputs, because it depends on a pre-trained language model. Besides, MetaPro is not a perfect system. The errors generated by MetaPro may also introduce biases for downstream applications.

References

- Ruchit Agrawal, Vighnesh Chenthil Kumar, Vigneshwaran Muralidharan, and Dipti Misra Sharma. 2018. No more beating about the bush: A step towards idiom handling for Indian language NLP. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*, pages 319–324.
- John A Barnden and Mark G Lee. 2002. An artificial intelligence approach to metaphor understanding. *Theoria et Historia Scientiarum*, 6(1):399–412.
- Erik Cambria, Qian Liu, Sergio Decherchi, Frank Xing, and Kenneth Kwok. 2022. SenticNet 7: A commonsense-based neurosymbolic AI framework for explainable sentiment analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3829–3839.
- Erik Cambria, Soujanya Poria, Alexander Gelbukh, and Mike Thelwall. 2017. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems*, 32(6):74–80.
- Lynne Cameron. 2003. *Metaphor in Educational Discourse*. A&C Black.
- Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. SemEval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval)*, pages 519–535.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.
- Huawen Feng and Qianli Ma. 2022. It’s better to teach fishing than giving a fish: An auto-augmented structure-aware generative model for metaphor detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 656–667.
- Mengshi Ge, Rui Mao, and Erik Cambria. 2022. Explainable metaphor identification inspired by conceptual metaphor theory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10681–10689.
- Mengshi Ge, Rui Mao, and Erik Cambria. 2023. A survey on computational metaphor processing techniques: From identification, interpretation, generation to application. *Artificial Intelligence Review*.
- James Geary. 2011. *I Is an Other: The Secret Life of Metaphor and How It Shapes the Way We See the World*. Harper Collins.
- Joseph Edward Grady. 1997. *Foundations of Meaning: Primary Metaphors and Primary Scenes*. University of California, Berkeley.
- Sooji Han, Rui Mao, and Erik Cambria. 2022. Hierarchical attention network for explainable depression detection on Twitter aided by metaphor concept mappings. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pages 94–104.
- Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225.
- George Lakoff. 1994. *Master Metaphor List*. University of California.
- George Lakoff and Mark Johnson. 1980. *Metaphors We Live by*. University of Chicago press.
- Chee Wee Ben Leong, Beata Beigman Klebanov, and Ekaterina Shutova. 2018. A report on the 2018 VUA metaphor detection shared task. In *Proceedings of the Workshop on Figurative Language Processing (FigLang)*, pages 56–66.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv e-prints*, pages arXiv–1907.
- Rui Mao and Xiao Li. 2021. Bridging towers of multi-task learning with a gating mechanism for aspect-based sentiment analysis and sequential metaphor identification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13534–13542.
- Rui Mao, Xiao Li, Mengshi Ge, and Erik Cambria. 2022. MetaPro: A computational metaphor processing model for text pre-processing. *Information Fusion*, 86-87:30–43.
- Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and WordNet based metaphor identification and interpretation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1222–1231.
- Rui Mao, Chenghua Lin, and Frank Guerin. 2019. End-to-end sequential metaphor identification inspired by linguistic theories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3888–3898.
- Rui Mao, Qian Liu, Kai He, Wei Li, and Erik Cambria. 2023. The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection. *IEEE Transactions on Affective Computing*.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The penn treebank. *Computational Linguistics*, 19(2):313–330.

- James H Martin. 1990. *A Computational Model of Metaphor Interpretation*. Academic Press Professional.
- James H Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension. *Trends in Linguistics Studies and Monographs*, 171:214.
- Srini Narayanan. 1997. Knowledge-based action representations for metaphor and aspect (KARMA). *Computer Science Division, University of California at Berkeley dissertation*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI*.
- John W Ratcliff and David E Metzener. 1988. Pattern matching: The gestalt approach. *Dr Dobbs Journal*, 13(7):46.
- Zachary Rosen. 2018. Computationally constructed concepts: A machine learning approach to metaphor interpretation using usage-based construction grammatical cues. In *Proceedings of the Workshop on Figurative Language Processing (FigLang)*, pages 102–109.
- Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a “kneedle” in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops (ICDCS-Workshops)*, pages 166–171. IEEE.
- Guangyao Shen, Jia Jia, Liqiang Nie, Fuli Feng, Cunjun Zhang, Tianrui Hu, Tat-Seng Chua, and Wenwu Zhu. 2017. Depression detection via harvesting social media: A multimodal dictionary learning solution. In *Joint Conference on Artificial Intelligence (IJCAI)*, pages 3838–3844.
- Ekaterina Shutova. 2010. Automatic metaphor interpretation as a paraphrasing task. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 1029–1037.
- Ekaterina Shutova. 2015. Design and evaluation of metaphor processing systems. *Computational Linguistics*, 41(4):579–623.
- Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A Method for Linguistic Metaphor Identification: from MIP to MIPVU*, volume 14. John Benjamins Publishing.
- Chuangdong Su, Fumiyo Fukumoto, Xiaoxi Huang, Jiyi Li, Rongbo Wang, and Zhiqun Chen. 2020. Deepmet: A reading comprehension paradigm for token-level metaphor detection. In *Proceedings of the Second Workshop on Figurative Language Processing (FigLang)*, pages 30–39.
- Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Neural metaphor detecting with cnn-lstm model. In *Proceedings of the Workshop on Figurative Language Processing (FigLang)*, pages 110–114.

DIAGRAPH: An Open-Source Graphic Interface for Dialog Flow Design

Dirk V \ddot{a} th*

Lindsey Vanderlyn*

Ngoc Thang Vu

University of Stuttgart, Germany

{vaethdk|vanderly|thang.vu}@ims.uni-stuttgart.de

Abstract

In this work, we present DIAGRAPH, an open-source¹ graphical dialog flow editor built on the ADVISER toolkit. Our goal for this tool is threefold: 1) To support subject-experts to intuitively create complex and flexible dialog systems, 2) To support rapid prototyping of dialog system behavior, e.g., for research, and 3) To provide a hands-on test bed for students learning about dialog systems. To facilitate this, DIAGRAPH aims to provide a clean and intuitive graphical interface for creating dialog systems without requiring any coding knowledge. Once a dialog graph has been created, it is automatically turned into a dialog system using state of the art language models. This allows for rapid prototyping and testing. Dialog designers can then distribute a link to their finished dialog system or embed it into a website. Additionally, to support scientific experiments and data collection, dialog designers can access chat logs. Finally, to verify the usability of DIAGRAPH, we performed evaluation with subject-experts who extensively worked with the tool and users testing it for the first time, receiving above average System Usability Scale (SUS) scores from both (82 out of 100 and 75 out of 100, respectively). In this way, we hope DIAGRAPH helps reduce the barrier to entry for creating dialog interactions.

1 Introduction

Dialog systems have gained much attention in recent years as they offer a convenient way for users to access information in a more personalized manner, or accomplish tasks through an intuitive natural language interface. Traditionally, they need to understand user input, track information across multiple dialog turns and choose a system response. These tasks can either be performed in a modular manner or as an end-to-end approach where user

*Both authors contributed equally

¹<https://github.com/DigitalPhonetics/diagraph>

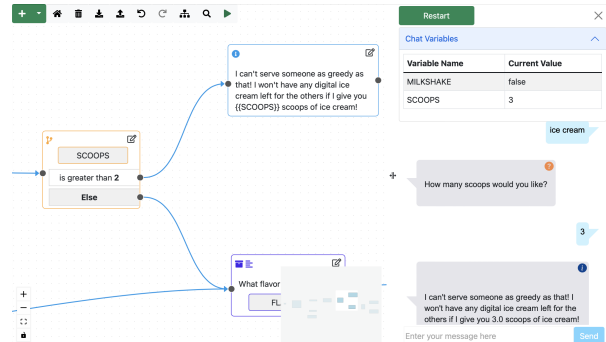


Figure 1: Left: Dialog editor with tutorial graph; Right: Debugging window with chat and variable explorer.

input is directly mapped to system output. However, regardless of approach, state-of-the-art approaches to dialog systems largely rely on neural methods (Chen et al., 2017). While these methods have generally shown improvements to dialog performance and generalizability across multiple dialog domains, they rely on the availability of sufficient training data which can be work-intensive and expensive to collect. Additionally, their decision-making often remains a black-box, which can make them unsuitable for highly sensitive domains, e.g., medical or legal contexts, where it is critical that dialog designers can maintain careful control over the system's behavior.

Although multiple toolkits have been developed to speed up the creation of dialog systems (Bohus and Rudnicky, 2009; Lison and Kennington, 2016; Ultes et al., 2017; Zhu et al., 2020; Li et al., 2020), and this has accelerated progress in the field, to the best of our knowledge all toolkits currently used for research require users to be able to write code in order to customize pre-implemented models for new domains or datasets.

However, this overlooks the fact that technical experts are not the only parties interested in creating dialog systems. Domain experts, or researchers from other disciplines, who may not have a technical background, may also be interested in using

dialog systems for a variety of goals. For example, domain experts might want to design a controlled production system. Researchers, e.g. psychologists or linguists, might want to design pilot studies or easily conduct research on interactions with a dialog system. Additionally, an interface to quickly setup data collection, e.g., to bootstrap an AI based dialog system or quickly iterate on user feedback, can accelerate dialog research or deployment.

To this end, we propose DIAGRAPH (figure 1): an open-source, graphical software for designing dialog flow graphs which are automatically converted into dialog systems. In this way, we hope it will serve as a good alternative to closed-source commercial options. Our goal for this tool is threefold: 1) To support subject-area experts to intuitively create and deploy complex and flexible dialog systems, 2) To support users, e.g., researchers, to easily and rapidly prototype and evaluate dialog systems or user behaviour, and 3) To provide a hands-on test bed for students learning about dialog systems. We evaluate DIAGRAPH with all three user groups and demonstrate its usability and practical usefulness by 1) Working with the department for business travel a University, 2) Performing a usability study where participants were asked to design or alter a dialog system in less than 30 minutes, and 3) Teaching a workshop on dialog systems and programming concepts to high school students.

2 Related Work

2.1 Dialog System Toolkits

In recent years, several toolkits have been developed to aid in the creation of dialog systems. Toolkits like RavenClaw (Bohus and Rudnicky, 2009), provide basic functionality, letting developers focus solely on describing the dialog task control logic. More recent toolkits include OpenDial (Lison and Kennington, 2016) – incorporating probabilistic rules and integration of external modules – and PyDial (Ultes et al., 2017) – a multi-domain dialog toolkit, for building modular dialog systems. Additionally, ConvLab (Lee et al., 2019; Zhu et al., 2020) and ADVISER (Ortega et al., 2019; Li et al., 2020) are modern toolkits, incorporating state of the art models in their implementations. In its recent update (Zhu et al., 2020), the developers also integrated evaluation and analysis tools to help developers debug dialog systems.

However, while these toolkits have accelerated dialog system research, their code-based interfaces

can be prohibitively complex for non-technical users and do not lend themselves to quick prototyping or education. While the ADVISER toolkit still uses a code-based interface for designing dialog systems, we chose to use it as the backend for DIAGRAPH due to the low overhead and flexibility of the toolkit.

2.2 Dialog Flow Designers

Recently, there has been research into efficiently navigating dialogs based on flow diagrams, such as work by Raghunathan et al. (2021), who investigate learning dialog policies from troubleshooting flowcharts and Shukla et al. (2020) who learn a dialog policy from a dialog flow tree designed using Microsoft’s graphic design tool. However, we are not aware of any well-fleshed out open-source tools for creating such graphs. Even though such dialog designer tools have become popular in industry with companies including Microsoft², Google³ and Amazon⁴ offering such services, the lack of open source alternatives impedes research around such graph/flow-based systems. Therefore, by providing a freely available alternative, we hope to make dialog system creation easier for researchers, domain experts, and educators.

To the best of our knowledge, the only open-source graphic-based dialog designer was created by Koller et al. (2018) as an educational tool to interface with Lego Mindstorms robots. While the authors show that it was well received by school and university students, its narrow scope does not address the needs of users such as subject-area experts or researchers. To this end, we create and publish DIAGRAPH: a general-purpose, open-source, graphical dialog designer built on the ADVISER toolkit.

3 Design Principles

The goal of the dialog designer is to allow for the intuitive creation of dialog systems with any level of technical knowledge. To this end, we design DIAGRAPH around the following principles:

User Friendliness To accommodate all three user groups, the software needs to be intuitive to operate without any previous programming experience. Thus, we try to keep interactions simple,

²<https://powervirtualagents.microsoft.com>

³<https://cloud.google.com/dialogflow?hl=de>

⁴<https://aws.amazon.com/de/lex/chatbot-designer/>

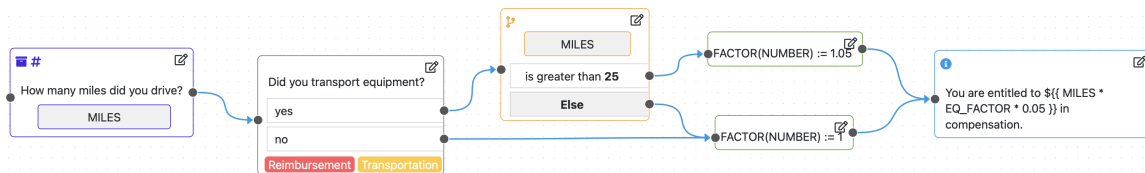


Figure 2: Example of each type of node. From left to right Variable Node (purple outline), User Response Nodes (black outline) Logic Node (orange outline), Variable Update Nodes (green outline), and Information Node (blue outline). The information node displays an example of the template syntax, using two variables in a mathematical expression to output a personalized message

e.g., dragging a connection from one node to the next to define dialog flow. We additionally try to only use icons and keyboard/mouse shortcuts commonly used in other programs, e.g., right clicking to get a context menu or typing `ctrl+f` to search. Finally, we include several features to help keep an overview even in complex graphs, e.g., a mini-map, text search and tags.

Flexibility To meet the needs of all user groups, we provide features for designing arbitrarily complex dialog systems. To personalize dialog outputs, we provide a template language which can be used to generate expressions based on previous user inputs and/or external data tables. To control dialog flow, we allow the storage of user inputs in variables, the creation of hidden variables, e.g. for loop counters, and blocks to split the dialog flow based on conditional logic.

Transparency Finally, we design DIAGRAPH to be transparent for all user groups, making it easy to understand the dialog structure and debug unintended behaviors. To this end, we provide a debugging view (figure 1) which can be opened in parallel to the dialog graph. Here, users can test out the different branches of their dialog as well as verify that the content of variables is correct at every turn. In this way, students can gain a deeper understanding of how the dialog system processes and researchers/subject-area experts can ensure that their dialog systems provide the correct outputs to end-users.

4 DIAGRAPH Software

DIAGRAPH is an open-source graphical software for designing dialog systems. The software consists of a web frontend and a python backend built on the ADVISER (Li et al., 2020) dialog system toolkit. DIAGRAPH enables users to design dialog systems by representing each turn as a node in a graph of the dialog flow. For each node, the dialog

designer can define the text which the dialog system will give to the end-user and where applicable, the possible end-user responses. Nodes can then be connected to form complex dialog flows. Each node or answer can only be connected to a single follow-up node, but a single follow-up node may be reached by multiple previous nodes. Additionally, cycles (loops) can be created by connecting a node to a node earlier in the graph, allowing for more complex dialog logic.

4.1 Nodes

The dialogs created with DIAGRAPH are built using five types of nodes (see Fig. 2), which can define even complex system behavior, such as storing variables, accessing data tables, and performing logical operations.

User Response Nodes are the fundamental building blocks of the dialog graph and allow branching dialog flow. These nodes provide a dialog system utterance and a finite set of possible end-user answer prototypes. During runtime, a node of this type expects end-user input, which will then be matched against the list of its answer prototypes. The prototype most similar to the end-user’s input will then be selected as the user intent, and the dialog will progress to the node connected to that answer. In case the intent recognition fails for some end-user inputs, designers may update a user response node to include answer synonyms that connect to the same follow-up node as the original answer prototype.

Information Nodes give information to the end-user without asking for input, acting as linear dialog flow. They are useful for presenting information, such as hints, to end-users when a decision from the end-user is not necessary. In this way, they can be used to split up long system answers into shorter, easier to read chunks to avoid overwhelming the end-user. Since Information Nodes

do not require end-user input, they can be directly connected to a single follow-up node.

Variable Nodes allow asking for user input and storing it in variables. They are similar to User Response Nodes in that they allow a dialog designer to define a system utterance and that they expect a response from the end-user. However, in contrast to User Response Nodes, the dialog designer does not define a set of prototypical end-user responses, but rather the general type of expected answer and the name of the variable which will store the value. Supported types include number, text, and Boolean. The user response is then stored inside this variable and can be used either to fill a template (see 4.1.1) or as part of more complex logical control. Variable nodes, like Information Nodes, can only be connected to a single follow-up node, but the values stored within the variable can be accessed at any point later in the dialog.

Variable Update Nodes are a way for dialog designers to either update the value of existing variables or create hidden variables which can be used to control the dialog flow, e.g., as loop counters. They do not provide output to the end-user.

Logic Nodes are purely used for dialog flow control, allowing to branch to follow-up nodes based on the values of variables, e.g. a number exceeding a certain threshold. This node does not provide any output to the end-user. Given a variable, Logic Nodes allow dialog designers to define a series of logical conditions based on the value of a variable. Each of the conditions can then be connected to a different follow-up node, personalizing the dialog based on previous end-user input. For more complex logic, Logic Nodes can also be connected to each other to define branches that require more than one condition or more than one variable.

4.1.1 Node Editor and Template System

In order to allow dialog designers to change the dialog system output, we incorporated the TinyMCE editor⁵. This allows dialog designers to visually format dialog system text, and include tables, links, or images. Additionally, the interface of TinyMCE is similar to that of common word processing software, reducing the barrier of such formatting.

We also provide a minimal template syntax (see Fig. 3) which can be used within a node to personalize system output based on previous input from an

⁵<https://www.tiny.cloud>

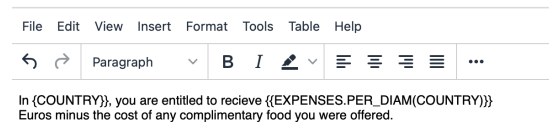


Figure 3: An example of the node editor, using the template syntax, which can be used to personalize output.

end-user and/or values from external tables which can be re-uploaded as information changes. Using the template syntax, for example, could allow dialog designers to create a single node, linked to an uploaded table which returns the per diem for a user based on the length of stay and the country of travel they have given. This can greatly simplify dialog graphs, both in terms of reducing the total number of nodes and answers needed (a single node instead of one for each country and duration) as well as in the ease of updating the graph as policies change (uploading a new table instead of editing nodes). The template syntax allows for the following operations, which can be combined together to create arbitrarily complex templates: 1) referencing the value of a variable, 2) performing mathematical operations, and 3) referencing a value from an uploaded table.

4.2 Navigation Features

As dialog graphs can become quite large for complicated domains, one important aspect of our tool is helping users to maintain an overview of the whole graph as well as to find individual nodes.

Mini-Map To keep track of how the section of graph they are working on at the moment fits into the bigger picture of the entire dialog, we provide a mini-map of the whole graph in the bottom right corner of the editor, highlighting the portion currently visible.

Search We provide a search panel with fuzzy matching to help find and update specific nodes in the graph. The user can click on a result to jump to the corresponding node, which will be highlighted and placed at the center of their screen. Additionally, as weblinks may change or need updating with more frequency than other types of information, we provide a similar panel where all weblinks are listed alphabetically.

Tags and Filtering Additionally, dialog designers can create tags and add them to any node in the dialog graph. Each tag will be assigned a color text and displayed as text at the bottom of the node

Name	# of Nodes	Sharable Link to Chat	Actions
Tutorial	57	Link	Edit Graph Rename Delete Log
CTS	1	Link	Edit Graph Rename Delete Log

[Create a new graph](#) [Feedback](#)

Figure 4: Graph management dashboard. Users can create/delete, edit, and share their dialog systems.

(see figure 2). The color coding helps to visually tell what category or categories a node belongs to which can help with grouping. Users can filter which tags are visible in the graph at a time, allowing them to hide graph sections not relevant to what they are currently working on.

4.3 Debugging

To test a dialog graph before releasing it, we provide a parallel debug window next to the dialog graph editor with an interactive instance of the corresponding dialog system. Dialog designers can then directly try out different inputs and verify whether the dialog flow works as intended. To simplify understanding of the dialog flow, the editor window will pan to and focus on the node currently displayed in the chat window. Any changes to the dialog flow in the editor will be immediately available in the chat, starting from the next dialog turn. Additionally, dialog designers can use the debug panel to view the values of all variables active in the dialog and ensure their correctness at every turn.

4.4 Managing Graphs

Dialog designers can manage their graphs from a central page 4. Here they can choose create new dialog systems or edit/delete existing ones. Additionally, they can download user interaction log data or get a link to their finished dialog system. This link, which can be distributed or embedded, points to a non-editable chat window (figure 5) where end-users can interact with the current version of the dialog system.

5 Implementation Details

5.1 Dialog System

Once the dialog graph has been defined, it can instantly be used as a fully functioning dialog system. The dialog system communication backend is based on the ADVISER toolkit (Li et al., 2020), which defines an abstract service class from which dialog modules can inherit. All modules which inherit from this service class can communicate with each other using a publish-subscribe framework. In

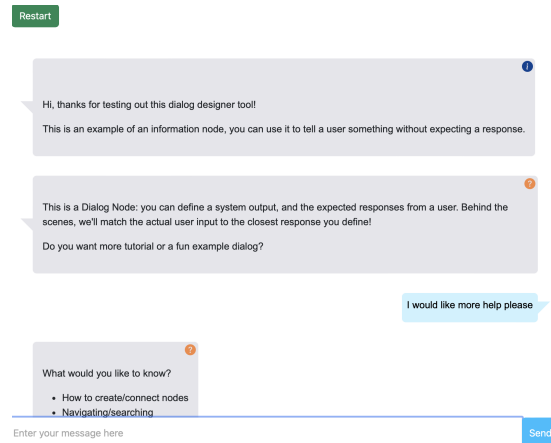


Figure 5: Embeddable/shareable chat window.

comparison to the communication protocol implemented in ADVISER, the backend of DIAGRAPH has been modified by attaching a user id to each message sent. In this way, the dialog systems created with DIAGRAPH can support an arbitrary number of end-users concurrently. The full system consists of two new modules: a policy and a natural language understanding unit (NLU). The policy navigates users through a dialog graph, choosing a next step based on the NLU output, which maps user input to one of the pre-defined answer candidates. As the dialog nodes themselves define the system output, there is no need for a natural language generation unit.

Currently, DIAGRAPH was designed for creating text-based dialogs. However, it would also be possible to create a spoken dialog system, e.g. by incorporating ADVISER’s text-to-speech and automatic speech recognition modules, which can communicate with the other DIAGRAPH services. This functionality is not included in the default DIAGRAPH distribution.

5.1.1 Natural Language Understanding

The natural language understanding unit (NLU) is based on a state-of-the-art, multilingual similarity model (Reimers and Gurevych, 2019) for User Response Nodes and regular expressions for Variable Nodes. As User Response Nodes have a fixed set of prototypical answers and the goal is to match the user input to one of these answers, a large language model performs well. For variable nodes, however, the input space must be restricted according to the variable’s type: e.g., a boolean variable should not be allowed to assume values other than *true* or *false*. Therefore, we use regular expressions to guarantee

that variable values conform to their specified type.

5.1.2 Dialog Policy

The dialog system’s behaviour, the policy, is primarily defined by the dialog designer based on how they construct their dialog graph. DIAGRAPH is by default configured to use a rules-based policy that will traverse the provided graph turn-wise, beginning from the start and taking the following actions depending on the current node type:

- **User Response Nodes:** The policy will output the system utterance, wait for user input, and then consider which prototypical answer best matches the user input (as determined by the NLU module). Finally, it will proceed to the node connected to that answer.
- **Information Node:** The policy will output the system utterance and transition to the connected node.
- **Variable Node:** The system will output the system utterance, wait for user input, store it in the associated variable, and then traverse to the connected node.
- **Variable Update Node:** The system will not output any text, but update the associated variable according to the specified rule. Then, it will move on to the connected node.
- **Logic Node:** The system will not output any text, but evaluate each logical condition based on the value of the associated variable. The system will then proceed to the node connected to the matched condition.

As an alternative to the handcrafted policy, the graphs generated from DIAGRAPH could be exported and directly used to train a reinforcement learning (RL) policy, as proposed in our earlier work (Väth et al., 2023). In comparison to the handcrafted policy included in the standard distribution of DIAGRAPH, the RL policy adapts to the amount of information in a given user query to either navigate the end-user through the dialog tree node by node or skip extraneous nodes once the user intent can be inferred.

5.2 Editor

The DIAGRAPH frontend is implemented in React, using the React Flow⁶ library to help smoothly render nodes and edges as they are moved around the

⁶<https://reactflow.dev>

screen. As graphs can be quite large, we focused on efficiency of rendering for the frontend, trying to keep both the amount of memory needed to run the editor and the amount of external libraries to a minimum. In this way, the editor can seamlessly support graphs with hundreds of nodes, in terms of creating or updating nodes, as well as in terms of fluid navigation. The dialog nodes created in the frontend are automatically stored in the backend database, which is updated every time information about the node (position, text, connections, associated answers, etc.) is changed. Keeping the frontend store and the backend database synced ensures that the handcrafted policy is always up to date with the state of the graph displayed in the editor. For handling the database connections, user authentication, (re)starting the dialog system, and serving a compiled version of the front end, we use the python toolkit Django⁷.

6 Evaluation

To evaluate the usefulness of our software, we tested its usability in three different scenarios 1) to create a real-world dialog system in a complex domain, 2) to rapidly prototype dialog systems, 3) to teach students about programming and dialog systems.

6.1 Complex Real-World Domain

As a first test, we worked with three subject-area experts from a university travel reimbursement department, to create a dialog system to help employees navigate travel planning and reimbursement. None of the experts had previous experience with chatbots, but hoped to offload common questions to the chatbot in order to have more time for complex cases and processing reimbursements.

Given the complexity of the domain and the importance of providing legally correct information, it served as a good test of DIAGRAPH’s full functionality. To create the dialog system, the subject-area experts first generated a set of frequently asked questions and then worked with us to sort them into categories and dialog sequences. Once they had a clear picture of how they wanted to group information, they were provided with an interactive tutorial and user manual for the system. After a collaborative phase to implement the first version of the dialog graph, the experts were left alone to expand and update it, resulting in a final version with

⁷<https://www.djangoproject.com>

194 nodes and a maximum depth of 32. The dialog system defined by this graph was then released for university employees as an additional option for answering travel related questions. During the initial test phase, approximately 2000 dialogs were conducted by university employees, each lasting roughly 5.9 turns

At the end of the collaboration, the experts were asked provide feedback about the experience via the System Usability Scale (Brooke, 1996), a ten item Likert scale for measuring user interfaces. They were also asked to give free-form feedback about their positive and negative impressions. DIAGRAPH received largely positive feedback with an average SUS score of 82 (highest possible 100; average 69 (Bangor et al., 2009)). Experts appreciated its user friendliness, how well dialogs could be specified, and the freedom for creative design the tool promoted.

This use-case highlights that the dialog designer can provide the level of control needed for highly complex and sensitive domains and be deployed in real-world scenarios.

6.2 Rapid Dialog System Design

In a second test, we investigated DIAGRAPH as a tool for rapidly creating dialog systems. We collected 19 participants and asked them to take 15-30 minutes to create and test a new dialog system of their own design. Participants were provided with a tutorial in the form of an interactive dialog graph and an example dialog of a digital ice cream seller. In contrast to the previous scenario, participants were not provided any in person instruction. Despite the lack of additional instruction and short interaction time, all participants were able to successfully develop a variety of dialog systems.

After interacting with DIAGRAPH, participants were also asked to fill out the 10 item SUS questionnaire and provide free-form feedback on things they liked or disliked about the tool. When evaluating the survey results, DIAGRAPH was given an average SUS score of 75 (out of a possible 100) indicating above average usability. This was also reflected in the comments, where the software was described as “fun!” and “intuitive to use”, although the tutorial dialog was generally seen as too long.

As results from the SUS can be considered to generalize when tested with at least 12 participants (Brooke, 2013), and because all users were able to create dialog systems in such a short time,

our results confirm that DIAGRAPH provides an intuitive interface which allows for rapid prototyping of dialog systems.

6.3 Educational Tool

Finally, we held a workshop on dialog systems with a group of six high school aged students to explore how DIAGRAPH could be used in an educational setting. Students were given a 45 minute long introduction to dialog systems and programming concepts. Following the theoretical introduction, they were given a 30 minute interactive tutorial – on how to create a chatbot for selling icecream with the DIAGRAPH tool – and then allowed to create and test their own dialog systems. The experience was rated fun and engaging by all participants (1.5 on a six-point Likert scale from very engaging to not at all engaging). All participants who left free-form feedback further indicated that they enjoyed the experience and/or felt that they learned a lot from it. Although not all participants had previous coding experience, all students were able to successfully create their own dialog graphs by the end of the half hour, each of which involved some type of logical operation or loop condition.

This experiment suggests that in addition to being easy to use, DIAGRAPH has potential as a teaching tool for dialog systems and for programming concepts.

7 Conclusion and Future Work

In this paper we have presented DIAGRAPH: an open-source graphic interface for designing dialog systems supporting either rules-based or RL-based dialog graph navigation. DIAGRAPH provides an intuitive way for subject-area experts to create complex dialog systems, users to rapidly prototype dialog interactions, and students to learn about dialog systems – regardless of the user’s level of technical background. Our user evaluation shows that DIAGRAPH was considered easy to use for all three use cases and users generally considered working with the tool an intuitive and fun experience.

In the future, we hope to extend our tool with the ability to query web APIs and to allow dialog designers to define expected inputs for variable nodes using custom regular expressions in order to increase flexibility even further. By releasing this software as open-source, we hope to make dialog design more accessible and to spark more research in controllable dialog policies.

References

- Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.
- Dan Bohus and Alexander I. Rudnicky. 2009. [The ravenclaw dialog management framework: Architecture and systems](#). *Computer Speech & Language*, 23(3):332–361.
- John Brooke. 1996. Sus—a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.
- John Brooke. 2013. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40.
- Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. [A survey on dialogue systems: Recent advances and new frontiers](#). *SIGKDD Explor. Newsl.*, 19(2):25–35.
- Alexander Koller, Timo Baumann, and Arne Köhn. 2018. DialogOS: Simple and Extensible Dialogue Modeling. In *Proc. Interspeech 2018*, pages 167–168.
- Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Zheng Zhang, Yaoqin Zhang, Xiang Li, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. 2019. [ConvLab: Multi-domain end-to-end dialog system platform](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 64–69, Florence, Italy. Association for Computational Linguistics.
- Chia-Yu Li, Daniel Ortega, Dirk Vāth, Florian Lux, Lindsey Vanderlyn, Maximilian Schmidt, Michael Neumann, Moritz Völkel, Pavel Denisov, Sabrina Jenne, Zorica Kacarevic, and Ngoc Thang Vu. 2020. [ADVISER: A toolkit for developing multi-modal, multi-domain and socially-engaged conversational agents](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 279–286, Online. Association for Computational Linguistics.
- Pierre Lison and Casey Kennington. 2016. [OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules](#). In *Proceedings of ACL-2016 System Demonstrations*, pages 67–72, Berlin, Germany. Association for Computational Linguistics.
- Daniel Ortega, Dirk Vāth, Gianna Weber, Lindsey Vanderlyn, Maximilian Schmidt, Moritz Völkel, Zorica Karacevic, and Ngoc Thang Vu. 2019. [ADVISER: A dialog system framework for education & research](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–98, Florence, Italy. Association for Computational Linguistics.
- Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, and Mausam. 2021. [End-to-end learning of flowchart grounded task-oriented dialogs](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4348–4366, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.
- Swadheen Shukla, Lars Liden, Shahin Shayandeh, Es-lam Kamal, Jinchao Li, Matt Mazzola, Thomas Park, Baolin Peng, and Jianfeng Gao. 2020. [Conversation Learner - a machine teaching tool for building dialog managers for task-oriented dialog systems](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 343–349, Online. Association for Computational Linguistics.
- Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young. 2017. [PyDial: A multi-domain statistical dialogue system toolkit](#). In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78, Vancouver, Canada. Association for Computational Linguistics.
- Dirk Vāth, Lindsey Vanderlyn, and Thang Vu Ngoc. 2023. Conversational tree search: A new hybrid dialog task. In *[forthcoming] Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. ACL Anthology.
- Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. [ConvLab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 142–149, Online. Association for Computational Linguistics.

disco: a toolkit for **D**istributional **C**ontrol of Generative Models

Germán Kruszewski*

Naver Labs Europe

firstname.lastname@naverlabs.com

Jos Rozen*

Naver Labs Europe

Marc Dymetman

Independent Researcher

marc.dymetman@gmail.com

Abstract

Pre-trained language models and other generative models have revolutionized NLP and beyond. However, these models tend to reproduce undesirable biases present in their training data. Also, they may overlook patterns that are important but challenging to capture. To address these limitations, researchers have introduced distributional control techniques. These techniques, not limited to language, allow controlling the prevalence (i.e. expectations) of any features of interest in the model’s outputs. Despite their potential, the widespread adoption of these techniques has been hindered by the difficulty in adapting the complex, disconnected code. Here, we present disco, an open-source Python library that brings these techniques to the broader public.¹

1 Introduction

The advent of pre-trained generative models has had a paradigm-shifting impact in Natural Language Processing (Radford et al., 2019b; Brown et al., 2020; Raffel et al., 2020), but also in other fields such as Speech Processing (Nguyen et al., 2022), Code Generation (Chen et al., 2021), Computer Vision (Ramesh et al., 2021; Rombach et al., 2022; Yu et al., 2022), among others. The common thread in these models is that of training a probability distribution over a given space of interest (text, images, audio, etc.) using large corpora, which can then be used to generate samples in this space. In particular, in NLP, these models have found applications not only in traditional tasks such as summarization (Radford et al., 2019b), but also opened new capabilities through few-shot learning (Brown et al., 2020). However, the models may suffer from deficiencies stemming both from replicating some patterns in the training data that are not desirable

* Equal contribution.

¹Available at <https://github.com/naver/disco>, and installable by `pip install disco-generation`. Demo video at <https://vimeo.com/800847322/9848219f33>.

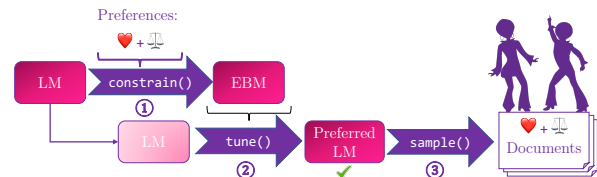


Figure 1: Overview of disco’s workflow.

such as offensiveness (Gehman et al., 2020) or unequal treatment (Cao et al., 2022), but also from failing to replicate other more desirable patterns which are also present in the data but are hard to capture by the neural network model, such as truthful information (Lin et al., 2022). For these reasons, there is a growing interest in controlling the generations to align with human values (Ouyang et al., 2022; Askell et al., 2021). Khalifa et al. (2021) proposed a comprehensive framework to tackle these issues that they coined “Generation under Distributional Control” or GDC. This framework builds on the idea introduced by Parshakova et al. (2019b) that we can decouple the problems of describing the target distribution representing the aligned generative model (i.e., the *what*) from the problem of approximating it (i.e., the *how*). In particular, they design the target distribution by fixing the desired expectations of some features of interest while avoiding catastrophic forgetting, and approximate it using the DPG algorithm (Parshakova et al., 2019a). Yet, other target distributions are possible. For example, Korbak et al. (2022b) showed that Reinforcement Learning from Human Feedback or RLHF (Ziegler et al., 2019; Bai et al., 2022; Ouyang et al., 2022) could also be framed as approximating a well-defined target distribution, highlighting the generality and flexibility of the distributional approach. Here, we present disco, a user-friendly library that provides developers, researchers, and practitioners easy access to state-of-the-art distributional control techniques. In what follows, we provide an overview of the GDC theoretical framework and its associated techniques

before introducing the toolkit, with an overview of some design choices and a quick tour of its capabilities. We then suggest possible applications and apply disco to three experimental use cases.

2 Background

Let’s assume a pre-trained generative model $a(\cdot)$ that defines a probability distribution over a sample space \mathcal{X} such that we can efficiently compute the probability $a(x)$ for any element $x \in \mathcal{X}$. Under the GDC framework, controlling the generative model amounts to defining a new probability distribution $p^*(x)$ to sample from. This probability distribution is such that **1. it meets the control conditions:** given a vector of n pre-defined real-valued functions (or *features*) $\phi(x) = [\phi_i(x)]_{i=1\dots n}$ defined over $x \in \mathcal{X}$, p^* is constrained such that each moment (i.e. expectation) $\mu_i \doteq \mathbb{E}_{x \sim p^*} \phi_i(x)$ matches a desired value $\bar{\mu}_i$; and **2. it avoids catastrophic forgetting:** p^* is the distribution that minimizes KL divergence from a among all distributions $p' \in \mathcal{C}$ satisfying the previous constraints $p^* \doteq \arg \min_{p' \in \mathcal{C}} D_{\text{KL}}(p', a)$. For example, if a is an English language model, $\phi_1(x)$ is a binary classifier detecting that a sentence topic is “sports” and $\phi_2(x)$ is another binary classifier that detects whether a sentence mentions a female character, and we set $\bar{\mu}_1 = 1$ and $\bar{\mu}_2 = 0.5$, then p^* will be a new language model that minimally deviates from a and such that all generated sentences speak about sports and 50% mention a female character.

[Khalifa et al. \(2021\)](#) show that p^* can be represented by an energy-based model (EBM) $P(x)$, i.e. a function that assigns a positive score to every x , such that $p^*(x) = P(x)/Z$ where $Z = \sum_{x \in \mathcal{X}} P(x)$. $P(x)$ can take either of the following two forms:

pointwise constraints: If we have binary features $\phi_i(x) \in \{0, 1\}$ and $\bar{\mu}_i = 1$, then,

$$P^{\text{point}}(x) = a(x) \prod_i \phi_i(x) \quad (1)$$

distributional constraints: More generally, we can express

$$P^{\text{distr}}(x; \lambda) = a(x) \exp(\lambda^\top \phi(x)). \quad (2)$$

where λ is a parameter vector of coefficients s.t. the resulting normalized distribution p^{distr} respects the desired constraints on the features’ moments. Finding the vector λ in Eq. 2 is done through a

training process by which λ is initialized to a random value, and then updated by gradient descent on minimizing $\mathcal{L}_{\text{coef}}(\lambda) = D_{\text{KL}}(p^*(\cdot), p^{\text{distr}}(\cdot; \lambda))$, with gradient

$$\nabla_{\lambda} \mathcal{L}_{\text{coef}}(\lambda) = \mathbb{E}_{x \sim p^{\text{distr}}(\cdot; \lambda)} \phi(x) - \bar{\mu} \quad (3)$$

and where the moments $\mathbb{E}_{x \sim p^{\text{distr}}(\cdot; \lambda)} \phi(x)$ are computed through self-normalized importance sampling (SNIS; [Owen, 2013](#)) using $a(\cdot)$ or any other proposal distribution ([Parshakova et al., 2019b](#); [Bengio and Senecal, 2008](#)).

2.1 Approximating p with an auto-regressive model

Once we have defined our target distribution p represented as an EBM P , we would like to use it for generation. Unfortunately, the EBM representation does not allow us to efficiently sample from it because it is not in an auto-regressive form. Yet, we can train an auto-regressive model π_{θ} to approximate $p(x) = P(x)/Z$ with the DPG algorithm ([Parshakova et al., 2019b](#)), which minimizes the forward KL divergence from the target distribution $D_{\text{KL}}(p, \pi_{\theta})$, or equivalently, the cross-entropy, obtaining the following gradient term:

$$\nabla_{\theta} \mathcal{L}_{\text{CE}}(\theta) = \frac{1}{Z} - \mathbb{E}_{x \sim q(\cdot)} \frac{P(x)}{q(x)} \nabla_{\theta} \log \pi_{\theta}(x). \quad (4)$$

Here $q(\cdot)$ is a distribution from which we can generate samples: We can set $q(x) = \pi_{\theta}(x)$ (on-policy version DPG_{on}), or alternatively use any other distribution (off-policy version DPG_{off}) (DPG; [Parshakova et al., 2019a](#)). The latter permits to improve the training stability by keeping a frozen version of π_{θ} as a proposal q and only update it when we are confident that $D_{\text{KL}}(p, \pi_{\theta})$ has improved (KL-adaptive DPG; [Khalifa et al., 2021](#)). Recently, [Go et al. \(2023\)](#) introduced f -DPG, which generalizes DPG to minimizing *any* f -divergence for approximating the target distribution. The family of f -divergences includes forward KL divergence, Jensen-Shannon, total variation distance (TVD), reverse KL, among others. Given a convex “generator” function f such that $f(1) = 0$, the gradient of the f -divergence $D_f(\pi_{\theta} || p)$ (in the on-policy version) is given by:

$$\nabla_{\theta} \mathcal{L}_f(\theta) = \mathbb{E}_{x \sim \pi_{\theta}} f' \left(\frac{Z \pi_{\theta}(x)}{P(x)} \right) \nabla_{\theta} \log \pi_{\theta}(x). \quad (5)$$

When $f'(t) = -\frac{1}{t}$, we recover the forward KL, $D_{\text{KL}}(p, \pi_\theta)$, objective of the original DPG algorithm. When $f'(t) = 1 + \log t$, we obtain an optimizer of the reverse KL, $D_{\text{KL}}(\pi_\theta, p)$. Alternatively, setting $f'(t) = \log \frac{2t}{t+1}$, recovers the gradient of the Jensen-Shannon divergence, and $f'(t) = \mathbb{1}[t > 1] - \frac{1}{2}$ yields the gradient of the total variation distance. Finally, we note that a constant ‘‘baseline’’ B can be subtracted from the $f'(Z\pi_\theta(x)/P(x))$ term in Eq. 5 to reduce the gradient’s variance (Korbak et al., 2022b; Go et al., 2023).

2.2 Further approximating p with Monte-Carlo sampling

Training the model π_θ in the above-described fashion can lead to a high-quality approximation of p but, often, it will not exactly match it. One way to further approximate the target distribution is to use quasi-rejection sampling (QRS; Eikema et al., 2022). This method consists in sampling from a proposal $q(x)$ (e.g., $q(x) \doteq \pi_\theta(x)$) and keeping only accepted samples with probability $\min(1, P(x)/(\beta q(x)))$, where β is a tunable parameter. The authors show that the f -divergence of the sampling distribution to the target distribution p is a monotonic function of β . In other words, increasing β can only improve (or maintain) the sampling fidelity, although at the cost of lower efficiency due to fewer accepted samples. Furthermore, they show that for any chosen β we can estimate the corresponding acceptance rate and divergence to p for any f -divergence.

2.3 Controlling conditional models

So far we have restricted our discussion to unconditional models. However, many NLP problems are modelled as *conditional* probability distribution $a(x|c)$ that takes some variable context c as input. Korbak et al. (2022a) proposed the following generalization of GDC to conditional models. They consider a distribution over contexts $\tau(c)$ and a map from a context c to a target EBM P_c with corresponding normalized distribution $p_c = P_c/Z_c$ where $Z_c = \sum_{x \in \mathcal{X}} P_c(x)$, which is respectively defined for **pointwise** and **distributional** constraints, as follows:

$$P_c^{\text{point}}(x) = a(x|c) \prod_i \phi_i(x, c), \quad (6)$$

$$P_c^{\text{distr}}(x|\lambda) = a(x|c) \exp(\lambda \cdot \phi(x, c)). \quad (7)$$

The model is then fine-tuned to optimize the loss function $\mathcal{L}_{\text{cond}}(\theta) = \mathbb{E}_{c \sim \tau} \text{CE}(p_c(\cdot), \pi_\theta(\cdot|c))$. Whereas Korbak et al. (2022a) only explored target distributions with pointwise constraints, for disco we also include distributional constraints. For this, we need to estimate the parameters λ , which we do by generalizing to the conditional case the derivation of Eq. 3:

$$\nabla_\lambda \mathcal{L}_{\text{coef}'}(\lambda) = \mathbb{E}_{c \sim \tau} \mathbb{E}_{x \sim p_c(\cdot|\lambda)} \phi(x, c) - \bar{\mu}. \quad (8)$$

2.4 RL with KL penalties

Another popular approach, seemingly competing with CDG, is Reinforcement Learning from Human Feedback or RLHF. This approach involves, first, learning a reward function $r(x)$ that approximates human judgments, and second, fine-tuning the model π_θ to maximize the reward while penalizing departure from the original $a(x)$.

$$J_{\text{RLKL}}(\theta) = \mathbb{E}_{x \sim \pi_\theta} \left[r(x) - \beta \log \frac{\pi_\theta(x)}{a(x)} \right]. \quad (9)$$

Interestingly, Korbak et al. (2022b) showed that this objective is equivalent to minimizing the *reverse* KL divergence to the target distribution:

$$p_{\text{RLHF}}(x) \propto a(x) \exp(r(x)/\beta). \quad (10)$$

Notably, Go et al. (2023) show that this target distribution could not only be approximated through the reverse KL divergence but also any other f -divergence, including forward KL and Jensen-Shannon, leading to different trade-offs in terms of expected reward and diversity. In particular, reverse KL tends to produce models with higher alignment levels as measured by the expected reward at the cost of lower diversity in the final model. On the other hand, the forward KL leads to lower alignment but preserving more diversity. Jensen-Shannon strikes a good balance in-between the two.

3 Design and implementation

disco is a Python toolkit based on PyTorch (Paszke et al., 2019) that abstracts away most of the details described in the previous section in a simple three-step workflow (Figure 1). It depends on the Transformers (Wolf et al., 2020) library, which allows it to load models seamlessly from the HuggingFace hub. The toolkit is organized around two fundamental classes of entities: **Samplers** and **Scorers** (see Figure 2). These entities are defined by exposing the methods `sample()` and `score()`, respectively.

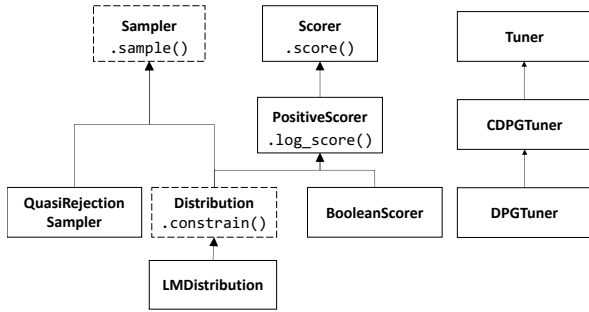


Figure 2: disco simplified class diagram. Dashed lines represent abstract entities.

As their name suggests, `sample()` draws samples from the underlying distribution, whereas `score()` computes a numerical score for each given sample. **PositiveScorers** are Scorers that are known to only return positive scores because of which they also provide the `log_score()` method. An entity can independently be a Sampler or a Scorer. However, we ask the generative models that we wish to control to support both the Sampler and the Scorer interface, further stipulating that the score of a sample corresponds to its sampling probability and is differentiable. We denote such classes **Distributions**. For example, a language model is encapsulated in an **LMDistribution** object, supporting both operations:

```
base = LMDistribution("gpt2")
samples, logprobs = base.sample()
samples_logprobs = base.log_score(samples)
```

`sample()` also returns `log_probs` that are consistent with `log_score()` for efficiency reasons.

Expressing preferences To express either pointwise or distributional preferences, Distributions support the `constrain()` method, which given a list of features $\phi_i(x)$ and their corresponding moments $\bar{\mu}_i$, returns a representation of the target distribution that respects the constraints while deviating minimally from the original model.² Features can be defined using the Scorer class, which accepts a function or a lambda abstraction taking a sample `s` and a context `c` as arguments and returning a score. An important class of features are *boolean* features, represented by the **BooleanScorer** class. While general features can only be used to define *distributional* constraints,

²The λ coefficients are approximately computed through importance sampling and SGD, both of which can be tuned by setting the number of samples employed and the SGD parameters when calling `constrain()`.

boolean features can also be used to define *pointwise* constraints. For example, we can score the presence of the string “amazing” in the sample `s`, as follows:

```
amazing = BooleanScorer(
    lambda s, c: "amazing" in s.text)
```

Conditional features can be expressed simply by taking the context `c` into account. Next, we can define an EBM with a *pointwise* constraint requiring that all our samples must include (the string) “amazing” by setting the target moment of a BooleanScorer feature to 1:

```
target = base.constrain([amazing], [1.0])
```

Distributional constraints are enforced by specifying any real-valued target moment or using non-binary features. Finally, an RLHF-like target distribution with regularization parameter `beta` and a reward scorer can be defined in the following way.

```
target = base * ExponentialScorer([reward],
    [1./beta])
```

In all cases, the resulting `target` is a **PositiveScorer** representing the target distribution as an EBM. Crucially, it is not an instance of **Distribution** since it does not allow sampling.

Fine-tuning the model To tune a **Distribution** to approximate the target EBM so that we can use it to generate samples, disco provides a set of **Tuner** classes, notably the **DPGTuner** and **FDPGTuner** for the unconditional case, and **CDPGTuner** and **FCDPGTuner** for the conditional case. The difference between using vanilla DPG/CDPG and *f*-DPG/*f*-CDPG for tuning is that whereas the former are restricted to minimizing the KL divergence to the target, *f*-(C)DPG can be used to minimize any *f*-divergence (defaulting to Jensen-Shannon, which often works well in practice).

```
model = LMDistribution("gpt2", freeze=False)
tuner = DPGTuner(model, target)
tuner.tune()
```

Note that we treat the unconditional case as a particular instance of the conditional one in which there is a single fixed context, the reason why (F)DPGTuner is also a (F)CDPGTuner. Conditional tuning only requires further specifying a distribution of possible contexts on which the model will be conditioned. This is done with a **ContextDistribution**, such as for instance

the `DatasetContextDistribution`, which samples contexts from HuggingFace Datasets (Lhoest et al., 2021). The Tuner reports a number of metrics that are useful to monitor the training progress. A number of **Logger** classes are provided to keep track of these metrics, including JSON, W&B, Neptune or custom loggers. One of the most important reported metrics includes the estimate of the model’s divergence to the target, `[kl/js/tv]_target_model`, as measured by KL, JS or TV, respectively, one of which is typically the quantity being optimized. Other metrics can include the features moments and the divergence from the base model if they are requested.

Improving the approximation with MC sampling After the tuning is done, `model` is now a better approximation to the target EBM, but it is not guaranteed to perfectly match this distribution. While further training can improve the situation, another alternative is using Quasi-Rejection Sampling (QRS; Eikema et al., 2022), a Monte-Carlo sampling technique that allows to trade-off sampling efficiency for a higher fidelity to the target distribution —a higher value of `beta` yields a better approximation at a higher computational cost by retaining a smaller fraction of samples.

```
sampler = QuasiRejectionSampler(
    target, model, beta=0.5)
samples, log_scores = sampler.sample()
```

The computational cost and the quality of the approximation will also depend on the proposal distribution (the closer the proposal is to the target, the higher quality can be obtained at a lower computational cost). Notably, we can estimate both quality in terms of divergence to the target or the expectation of a feature of interest and computational cost in terms of the expected acceptance rate for any given proposal distribution and value of `beta`:

```
estim = QuasiRejectionSamplerEstimator(
    target, model)
beta = 0.5
ar_at_beta = estim.acceptance_rate_at_beta(beta)
kl_at_beta = estim.divergence_at_beta(beta, KL)
amazing_at_beta = estim.feature_moment_at_beta(
    beta, amazing)
```

4 Applications

`disco` enables a number of possible applications, of which here we list only a few.

Compilability/style constraints on code generation Language models trained on clean code data

can still generate code that does not compile or, even if it does, can fail to meet style standards. Korbak et al. (2021, 2022a) showed that it was possible to effectively improve code generation models on both accounts by using pointwise constraints on the result coming from the Python compiler and of an off-the-shelf linter.

Limiting hallucinations Seq2seq models such as those used in summarization or NMT have a common failure mode by which they generate information not originally present in the source document (aka “hallucinations”). Entity-level factual consistency (Nan et al., 2021) is a family of measures that detect whether produced entities were included in the source, and whether they are part of the target in the dataset. Korbak et al. (2022a) showed that GDC could be successfully applied to improve on these metrics. Below, we reproduce part of the experiments.

Debiasing language models GDC can address bias in language models by defining a feature detecting a population of interest, and setting the target moments of the feature to the desired value. Khalifa et al. (2021) experimented with reducing gender bias, while Go et al. (2023) use this technique to balance the “regard” score among different religious groups.

5 Showcase experiments

This section presents a selection of experiments to showcase a few use cases of `disco`, along with code snippets illustrating their implementation.

5.1 Amazing experiment

In this simple experiment, initially introduced in Khalifa et al. (2021), we want *all* samples from the GPT-2 (small) language model (Radford et al., 2019a) to contain the string “amazing”. The following code shows how to tackle this task in `disco`. We experiment with different batch sizes (`n_samples_per_step` $\in \{2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\}$) while controlling the total number of gradient steps (`n_gradient_steps` $\in \{2^5 \times 1000, 2^4 \times 1000, 2^3 \times 1000, 2^2 \times 1000, 2^1 \times 1000, 2^0 \times 1000\}$) so that the total number of samples remains constant. `sampling_size` and `scoring_size` only affect speed and are set to the maximum value that fits in the GPU memory.

```
base = LMDistribution("gpt2", device="cuda")
amazing_scorer = BooleanScorer(
```

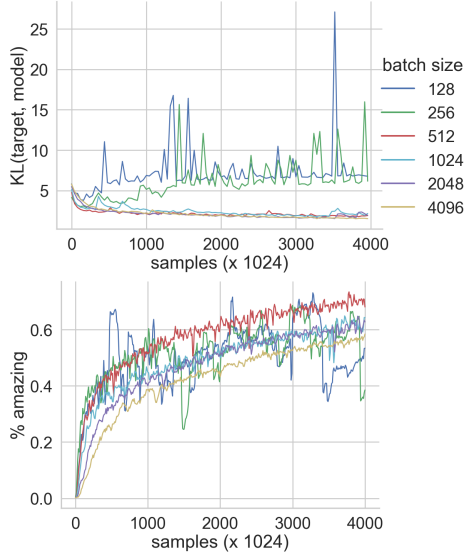


Figure 3: Divergences to the target distribution (top) and proportion of “amazing” samples during tuning (bottom), for various batch sizes.

```

lambda s, c: "amazing" in s.text)
target = base.constrain(
    [amazing_scorer], [1])
model = base.clone().freeze(False)

tuner = DPGTuner(model, target,
    n_gradient_steps=1000,
    n_samples_per_step=4096,
    sampling_size=64,
    scoring_size=64)
tuner.tune()

```

Results Figure 3 shows the KL divergence of the model to the target distribution (top) and the proportion of sequences containing “amazing” (bottom). The former is the optimized metric, subsampling the percentage of “amazing” sequences and, importantly, the divergence from the original distribution. Although small batch sizes seem to give good enough results for the “amazing” feature, their divergences are almost off the chart, indicating model degradation. On the other hand, the model trained with batch size 4096 has a KL of 1.47 nats and generates “amazing” samples 57% of the time (from an initial frequency of about 0.1%). Additionally using QRS ($\beta = 0.02$) retains just 10% of the samples, but gets us to 0.08 nats and generates 100% “amazing” samples. For illustrative purposes, some randomly generated samples are presented in the Appendix.

5.2 Don’t hallucinate entities

Here we replicate the setting described in Korbak et al. (2022a) on improving entity-level factual consistency (Nan et al., 2021). Specifi-

cally, we constrain a T5 (small) model (Raffel et al., 2019) so that all named entities appearing in the summary also appear in the source, with at least 4 entities appearing in the summary. Given a function $\text{NER}(x)$ that returns a set of named entities implemented with the SpaCy (Honibal et al., 2020) pre-trained named entity recognizer, we build two features: `no_new_entity`, and `min_four_entities`, which given a sample x and a context c , compute $\text{NER}(x) \subseteq \text{NER}(c)$ and $|\text{NER}(x)| \geq 4$, respectively. We train using a CDPGTuner that samples *source* documents from the first 5k documents in the CNN / DailyMail (Nallapati et al., 2016) dataset, via a `DatasetContextDistribution`.

```

base = LMDistribution("t5-small",
    auto=AutoModelForSeq2SeqLM, device="cuda")
target = base.constrain(
    [no_new_entity, min_four_entities],
    [1, 1])
model = base.clone().freeze(False)

contexts = DatasetContextDistribution(
    dataset="cnn_dailymail", subset="1.0.0",
    split="train[:5000]", key="article",
    prefix="summarize: ")
tuner = CDPGTuner(model, target,
    context_distribution=contexts,
    n_gradient_steps=1000,
    n_samples_per_step=32,
    context_sampling_size=32,
    sampling_size=8,
    scoring_size=8)
tuner.tune()

```

Results We use beam search to sample summaries x for source documents c in the test set. Their entity-level factual consistency, measured by precision to the source ($|\text{NER}(x) \cap \text{NER}(c)| / |\text{NER}(c)|$), improves from .91 to .94, and recall to the target t ($|\text{NER}(x) \cap \text{NER}(t)| / |\text{NER}(t)|$) goes from .26 to .45. Notably, the summaries’ ROUGE-L score also slightly improves, from 0.257 to 0.268. Example summaries are presented in the Appendix.

5.3 The entertainer

In this experiment we want to control the personality type of a BlenderBot (Roller et al., 2021) chatbot according to Myers&Briggs dimensions (Myers and Myers, 1995) (Extraverted/Introverted, iNtuitive/obSservant, Thinking/Feeling, Judging/Prospecting), targeting a “spontaneous and generous” ESFP³ type. Specifically, we use a pre-trained classifier to assess personality types⁴

³<https://www.16personalities.com/esfp-personality>

⁴<https://huggingface.co/spaces/seduerr/personality>

Responses to “What’s the best piece of advice you’ve ever been given?”	E	S	F	P
<i>before tuning</i>				
mine is staying confident. It’s tough though when I dont really have advice sometimes	0.6	0.36	0.62	0.34
There’s probably so many. I love helping people get better. By giving them information and securing they can better themselves	0.48	0.24	0.47	0.62
<i>after tuning</i>				
Human beings do not belong to a single continent	0.86	0.84	0.72	0.5
I’d have to say knowledge and dedication are definitely what keep me from failing.	0.64	0.76	0.8	0.65

Table 1: Personality Type ESFP score for BlenderBot’s samples, before and after tuning

and built a `PersonalityTypeScorer` that returns the score of any chosen dimension. We use the `facebook/blenderbot-400M-distill-seq2seq` model from the HuggingFace hub. We set the target moments to 0.8 on each of the “E”, “S”, “F”, and “P” personality dimensions. To prompt the model with relevant context, we use a list of “icebreaking” utterances collected from the web⁵ to build a `ContextDistribution`, which is used both when estimating the coefficients of the EBM and for fine-tuning the model using a `CDPGTuner`.

```
base = LMDistribution(
    "facebook/blenderbot-400M-distill",
    auto=AutoModelForSeq2SeqLM,
    device="cuda")
contexts = ContextDistribution(
    "data/icebreakers.txt")
target = base.constrain(
    [PersonalityTypeScorer(t)
     for t in "ESFP"], [0.8] * 4,
    context_distribution=contexts)
model = base.clone().freeze(False)

tuner = CDPGTuner(model, target,
    context=contexts,
    n_gradient_steps=2000,
    n_samples_per_step=512,
    context_sampling_size=8,
    sampling_size=128,
    scoring_size=128)
tuner.tune()
```

Results We improve the moments of the dimensions of interest, as follows: E: .59 \rightarrow .64, S: .42 \rightarrow .56, F: .55 \rightarrow .69, P: .48 \rightarrow .56. Some samples are shown in Table 8 and in the Appendix.

6 Related works & Conclusion

`disco` is the first toolkit to bring GDC techniques to a wide audience. Such techniques build on a solid theoretical framework based on the separation between the design of the target distribution and its

⁵<https://museumphack.com/list-icebreakers-questions>

approximation. Thanks to this elegant approach, users can first focus exclusively on defining the control conditions by setting the desired expectations of features of interest. Then, they can use the tools provided in `disco` (like the f -(C)DPG and the QRS algorithms) to generate content meeting the desired conditions. Notably, GDC subsumes other frameworks such as RLHF, which can be seen as a particular case (see Sec. 2.4). For this reason, `disco` has a wider scope than other related toolkits such as RL4LM (Ramamurthy et al., 2022), which centers on RL methods only. Nevertheless, there is a large space for cross-pollination between RL-based frameworks and `disco` because of similarities in the underlying algorithms (Korbak et al., 2022b). For example, `disco` incorporates the baseline technique from RL to reduce the gradients’ variance and increase training stability and efficiency. Likewise, there are many points of contact between the two paradigms that remain to be explored in the future which can further enhance `disco`.

Acknowledgements

We thank Tetiana Parshakova, Hady Elsahar, Muhammad Khalifa, Bryan Eikema and Tomasz Korbak for earlier contributions that helped shape `disco`. We also thank Ronald Cardenas for testing parts of the library.

Broader impact

The techniques made broadly accessible by `disco` have the potential to address many existing challenges of language models and other generative systems such as bias, factual consistency, toxicity, just to name a few. `disco` is a very general framework that allows to control the prevalence of any feature that can be represented as a function from a sample to a numerical score (for example, a classifier’s score, a reward function or any other metric of the text). Because of this generality `disco` can adapt to a wide range of use cases and changing values and demands. However, the concrete results will depend on how the controlled features are quantified, on which `disco` is completely unopinionated. The crucial work of deciding how to best design relevant features and their target moments is a task the user will have to undertake. On the other hand, the users now have the power to focus exclusively on this latter question and relegate the algorithmic problems of controlling the model to match their desiderata to `disco`.

References

- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. [A general language assistant as a laboratory for alignment](#). *ArXiv preprint*, abs/2112.00861.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#).
- Yoshua Bengio and Jean-Sébastien Senécal. 2008. [Adaptive importance sampling to accelerate training of a neural probabilistic language model](#). *IEEE Trans. Neural Networks*, 19(4):713–722.
- 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 *Proc. of NeurIPS*.
- Yang Cao, Anna Sotnikova, Hal Daumé III, Rachel Rudinger, and Linda Zou. 2022. [Theory-grounded measurement of U.S. social stereotypes in English language models](#). In *Proc. of NAACL-HLT*, pages 1276–1295, Seattle, United States. Association for Computational Linguistics.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. [Evaluating large language models trained on code](#). *ArXiv preprint*, abs/2107.03374.
- Bryan Eikema, Germán Kruszewski, Christopher R Dance, Hady Elsahar, and Marc Dymetman. 2022. [An approximate sampler for energy-based models with divergence diagnostics](#). *Transactions on Machine Learning Research*.
- Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. [RealToxicityPrompts: Evaluating neural toxic degeneration in language models](#). In *Findings of EMNLP*, pages 3356–3369, Online. Association for Computational Linguistics.
- Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. 2023. [Aligning language models with preferences through f-divergence minimization](#). *ArXiv preprint*, abs/2302.08215.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. [A distributional approach to controlled text generation](#). In *Proc. of ICLR*. OpenReview.net.
- Tomasz Korbak, Hady Elsahar, Marc Dymetman, and Germán Kruszewski. 2021. [Energy-based models for code generation under compilability constraints](#). *ArXiv preprint*, abs/2106.04985.
- Tomasz Korbak, Hady Elsahar, German Kruszewski, and Marc Dymetman. 2022a. [Controlling conditional language models without catastrophic forgetting](#). In *Proceedings of ICML*, pages 11499–11528. PMLR.
- Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. 2022b. [On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting](#). In *Proc. of NeurIPS*.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. [Datasets: A community library for natural language processing](#). In *Proc. of EMNLP*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. [TruthfulQA: Measuring how models mimic human falsehoods](#). In *Proc. of ACL*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.
- Isabel Briggs Myers and Peter B. Myers. 1995. *Gifts differing: understanding personality type*, first edition edition. Davies-Black Publishing, Palo Alto, California.
- 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.

- Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero Nogueira dos Santos, Henghui Zhu, Dejjiao Zhang, Kathleen McKeown, and Bing Xiang. 2021. [Entity-level factual consistency of abstractive text summarization](#). In *Proc. of EACL*, pages 2727–2733, Online. Association for Computational Linguistics.
- Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoit Sagot, Abdelrahman Mohamed, et al. 2022. [Generative spoken dialogue language modeling](#). *ArXiv preprint*, abs/2203.16502.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Proc. of NeurIPS*.
- Art B. Owen. 2013. [Importance Sampling](#). In *Monte Carlo theory, methods and examples*, chapter 9. Unpublished Lecture Notes.
- Tetiana Parshakova, Jean-Marc Andreoli, and Marc Dymetman. 2019a. [Distributional reinforcement learning for energy-based sequential models](#). *ArXiv preprint*, abs/1912.08517.
- Tetiana Parshakova, Jean-Marc Andreoli, and Marc Dymetman. 2019b. [Global autoregressive models for data-efficient sequence learning](#). In *Proceedings of CoNLL*, pages 900–909, Hong Kong, China. Association for Computational Linguistics.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In *Proc. of NeurIPS*, pages 8024–8035.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019a. Language models are unsupervised multitask learners.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019b. 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. 2019. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#).
- 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. [Is reinforcement learning \(not\) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization](#). volume abs/2210.01241.
- Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. [Zero-shot text-to-image generation](#). In *Proc. of ICML*, volume 139, pages 8821–8831. PMLR.
- 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 *Proc. of EACL*, pages 300–325, Online. Association for Computational Linguistics.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of CVPR*, pages 10684–10695.
- 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 *Proc. of EMNLP*, pages 38–45, Online. Association for Computational Linguistics.
- Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. 2022. [Scaling autoregressive models for content-rich text-to-image generation](#). *Transactions on Machine Learning Research*. Featured Certification.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. [Fine-tuning language models from human preferences](#). *ArXiv preprint*, abs/1909.08593.

A Additional example generations

GPT-2 base model	
×	'I just read the information that this made a lot of people feel so ashamed they hadn't heard anything about the existing government and I'm very proud of that,' he said. 'I believe it
×	Reasons for showing up further after guessing. Fiber optic cables could be cheap to deploy but at least they cover a lot of ground. VR Fitness on Facebook:\n\nI've started modeling a concept
×	Like every other generation is awash in data, from sedan cars to it's internal combustion engine, we have a new problem with the world: Lattes fluctuate from year to year because of
×	Write down every magic stone found in the cave. Decide what level to take. 1) Find Adamantite orders in what spell the game sets up 1) Name yourself spell numbers 2) Discover further
×	\nPosted by Soumitai on May 01, 2015 at 7:57 am Permalink\n\nRating\n\n1 Related\n\nMust Someone Build It There is no way ANONYMOUS
×	By choosing a brand name - including Audi or Powerleague, DeMarcus Atlanta's full name should go with national advertising.\n\nAron's fifth, retiring Chicago Bulls future Hall of Famer is
×	Gothic Witch Queens: Hound of Innocence Acts 1 Acheryl Solde - Thieves Don't Approach Me Act, The monsters are gone, Sonofie, The Ghost Just Broke the
×	In the interview before the Charlottesville rally, he argued for those who oppose arming the so-called alt-right:\n\nFirst of all, I happen to just strongly disagree with the anti-Second
×	Sophie Pottle has been playing guitar and lead singer of the band Zoey Emaleigh since 2008. Cast in the role of Flaming Bob this year, Jayne has very less of
×	'He could have died,' said the ex-German intelligence officer in his mind. 'He didn't want to be hunted, or forced into liberty.' Other former German officers had once and legendary former

Table 2: Random samples extracted from the gpt2 base model, scored for whether the string “amazing” is present or not.

Fine-tuned model with DPG to include “amazing” (batch size 4096)

- ✓ Say thanks by giving speetuh323 a tip and help them continue to share **amazing** Things with the Thingiverse community.\n\nStep 1:<|endoftext|>
 - × ers should consider FB2; a DB2, and x3.\n\nThis build is essentially a 4 full run DB2 (see later post) with some pretty good changes to see how easy
 - ✓ 1022) October 7, 2016\n\nIt's **amazing**. As **amazing** as it might have been, the undefeated Boston College Utes do do this despite the Hahn-1 & Carlston refs more
 - ✓ **Amazing** Review.\n\n1 outline of design shown below\n\nTo conclude Was their first (pun intended). Cards look alike really No issues with the endosing front. And the car itself looks nice
 - ✓ **amazing** - 4 of this.\n\nUpdate: They have added some changes to update your ~/.**amazingrc** file\n\nAnyway.\n\nAnd they can detect your smoth status, too'
 - ✓ It was **amazing** how transformed Caitlyn is at the moment in Stephane's third birthday. I led her to make an **amazing** face at the election.\n\n\spverbThe people closest to her had no idea it was
 - ✓ For pretty bad reason, we've been hearing a lot about the unlevel GameClip for a while... spending weeks whipping up info in bulk, sharing **amazing** new features, and playing it to nuts
 - × One of the things that wobble around town when I use it is that I sometimes end up being incredibly bad at explaining to others why they're doing it. So here's what I've learned,
 - ✓ Artwork and design are equally important and thus perplexing. I always don't create boring design for anyone that can't go otherwise around the office. An **amazing** pair of shoes because of their delicate detailing
 - × a clearly beneficial single meaning. It began in five minutes with ones of Neil de Grasse Tyson, which is surprising, given where he went from here. Here it comes in three steps:\n\n
-

Table 3: Random samples extracted from the model fine-tuned with DPG (batch size 4096) on the objective that all sequences should contain “amazing” while minimally diverging from the origin

QRS (beta=0.02) with a DPG-tuned proposal (batch size 4096)

- ✓ Sandbox 20 exclusive\n\nThursday, May 5, 2011 Main Theme - Masjidn 14 A probably **amazing** theme that is pretty memorable. And it's shipped with 20 coolest and last exclusive skins
 - ✓ "Remember the Did Ya Wanna Look So Good?" that Randall offered to Griffin and Cartman E'\nBrien. "Remember the **amazing** reveille you brought my friends?"\n\nGod bless them.
 - ✓ 500 years ago, Times Mouse was celebrated for notasting and giving minrs their gift birds from America.\n\nWith "Ten Thousand Saints," Times Mouse **amazingly** took holiday every year since 1983 now that
 - ✓ GODNS\n\n"Free love is an **amazing** truth." -President Franklin D. Roosevelt\n\nGODNAMES\n\nCares about the other., Astonishing.\n\nCONGRE
 - ✓ Viticos Crystallographie is now available as an experimental 8ish compendium.\n\nREAD MORE >>\n\nThe last chance, in the last few years & **amazingly** beautiful, at doing
 - ✓ I know I missed out on the **amazing** (or at least a little impressive!) gradient experience, but here it is in action. It'll make either seat you taller or shorter and seems pretty much synchronized
 - ✓ Can Brewing Company Bottle Collection hold up?\n\n\nSince back in 2007 we have been sharing **amazing** Tank series for some of our styles out in the world:\n\nBig Barrel Wit - A range of
 - ✓ Cast & Crew Episode 77 Welcome to Cast & Crew Episode 77. This 44 minute podcast brings we funny figures and some great hosts like on but very lovable dreams. Featuring Ghostbusters have had **amazing** paydays
 - ✓ Honey! It is absolutely **amazing**!! in a whole good way! People are not talking so much about, you know, strawberries, health check, growing organ. I'm signing off. It's
 - ✓ There are perks and payments for top players and promotions: we can rest assured that you will agree to receive us all you want in addition to our **amazing** Golden Key Card, VIP offsite events, contests
-

Table 4: Random samples extracted using QRS with parameter beta=0.02 on the target distribution in which all sequences contain the string “amazing” using a DPG fine-tuned model (batch size 4096) as a proposal.

Original article

(CNN) Seventy years ago, Anne Frank died of typhus in a Nazi concentration camp at the age of 15. Just two weeks after her supposed death on March 31, 1945, the Bergen-Belsen concentration camp where she had been imprisoned was liberated – timing that showed how close the Jewish diarist had been to surviving the Holocaust. But new research released by the Anne Frank House shows that Anne and her older sister, Margot Frank, died at least a month earlier than previously thought. Researchers re-examined archives of the Red Cross, the International Training Service and the Bergen-Belsen Memorial, along with testimonies of survivors. They concluded that Anne and Margot probably did not survive to March 1945 – contradicting the date of death which had previously been determined by Dutch authorities. In 1944, Anne and seven others hiding in the Amsterdam secret annex were arrested and sent to the Auschwitz-Birkenau concentration camp. Anne Frank's final entry . That same year, Anne and Margot were separated from their mother and sent away to work as slave labor at the Bergen-Belsen camp in Germany. Days at the camp were filled with terror and dread, witnesses said. The sisters stayed in a section of the overcrowded camp with no lighting, little water and no latrine. They slept on lice-ridden straw and violent storms shredded the tents, according to the researchers. Like the other prisoners, the sisters endured long hours at roll call. Her classmate, Nannette Blitz, recalled seeing Anne there in December 1944: "She was no more than a skeleton by then. She was wrapped in a blanket; she couldn't bear to wear her clothes anymore because they were crawling with lice." Listen to Anne Frank's friends describe her concentration camp experience . As the Russians advanced further, the Bergen-Belsen concentration camp became even more crowded, bringing more disease. A deadly typhus outbreak caused thousands to die each day. Typhus is an infectious disease caused by lice that breaks out in places with poor hygiene. The disease causes high fever, chills and skin eruptions. "Because of the lice infesting the bedstraw and her clothes, Anne was exposed to the main carrier of epidemic typhus for an extended period," museum researchers wrote. They concluded that it's unlikely the sisters survived until March, because witnesses at the camp said the sisters both had symptoms before February 7. "Most deaths caused by typhus occur around twelve days after the first symptoms appear," wrote authors Erika Prins and Gertjan Broek. The exact dates of death for Anne and Margot remain unclear. Margot died before Anne. "Anne never gave up hope," said Blitz, her friend. "She was absolutely convinced she would survive." Her diary endures as one of the world's most popular books. Read more about Anne Frank's cousin, a keeper of her legacy .

Base T5 summary

typhus is an infectious disease caused by lice that breaks out in places with poor hygiene. a deadly typhus outbreak caused thousands to die each day. typhus is an infectious disease caused by lice that breaks out in places with poor hygiene.

Fine-tuned T5 summary

Anne Frank and her older sister, Margot, died at least a month earlier than previously thought. researchers re-examined archives of the Red Cross, the International Training Service and the Bergen-Belsen Memorial. they concluded that Anne and Margot probably did not survive to March 1945.

Table 5: Summaries generated using beam search with beam size 5 from the T5-small model and from the one fine-tuned with the objective of producing at least 4 named entities that are in the source document, as described in Section 5.2. Highlighted in purple are the named entities in the text recognized by SpaCy.

Original article

(CNN)The **FBI** charged a **Philadelphia** woman on Thursday with trying to travel overseas to fight for **ISIS**. She's one of three women arrested this week on terror charges. Two **New York** women were also taken into custody. An **FBI** complaint cites numerous social media messages dating back to August 2013 that were sent by Keonna **Thomas**, 30, also known as "Young Lioness" and "Fatayat **Al Khilafah**." One Twitter message said, "If we truly knew the realities ... we all would be rushing to join our brothers in the front lines pray ALLAH accept us as shuhada [martyrs]." Another said, "When you're a mujahid [violent jihadi fighter] your death becomes a wedding." The **FBI** said **Thomas** purchased an electronic visa to **Turkey** on March 23. **Turkey** is known as the easiest place from which to enter **Syria** and join **ISIS**. An **ISIS** manual advises recruits to buy round-trip tickets to vacation spots such as **Spain** and then purchase tickets for their real destination once they arrive overseas, the **FBI** said. On March 26, **Thomas** purchased a ticket to **Barcelona**, with a March 29 departure and an April 15 return to **the United States**, the complaint said. It's not clear when or where she was arrested. She was charged with knowingly attempting to provide material support and resources to a designated foreign terrorist organization. She could be sentenced to 15 years in prison. On Thursday, **Noelle Velentzas**, 28, and her former roommate, **Asia Siddiqui**, 31, were arrested in **New York** and accused of planning to build an explosive device for attacks in **the United States**, federal prosecutors said. In the past 18 months, **the Justice Department's National Security Division** has prosecuted or is prosecuting more than 30 cases of people attempting to travel abroad to join or provide support to terrorist groups. Of those cases, 18 allegedly involve support to **ISIS**. "The terrorist threat is more decentralized, more diffuse, more complicated," **Homeland Security** Secretary **Jeh Johnson** told reporters Thursday. "It involves the potential lone wolf actor, it involves the effective use of social media, the Internet."

Base T5 summary

a woman is charged with trying to travel overseas to fight for **ISIS**. she's one of three women arrested this week on terror charges. two **new york** women were also taken into custody.

Fine-tuned T5 summary

the **FBI** charged a **Philadelphia** woman with trying to travel overseas to fight for **ISIS**. **Keonna Thomas**, 30, also known as "young **Lioness**" and "Fatayat **Al Khilafah**" two **new york** women were also taken into custody.

Table 6: Summaries generated using beam search with beam size 5 from the T5-small model and from the one fine-tuned with the objective of producing at least 4 named entities that are in the source document, as described in Section 5.2. Highlighted in purple are the named entities in the text recognized by SpaCy.

(CNN)President **Barack Obama** tied himself to the mast of a nuclear deal with **Iran** even before he became the **Democratic** candidate for president. Reaching a good, solid agreement with **Iran** is a worthy, desirable goal. But the process has unfolded under the destructive influence of political considerations, weakening **America's** hand and strengthening **Iran**. **Obama's** political standing and his historic legacy in foreign policy are so deeply intertwined with reaching an accord with **Iran** that if the deal ultimately collapses, he may fear that historians will conclude that his legacy in global affairs collapsed with it. There is a reason one gets the feeling that it is **the United States** and not **Iran** that is the more eager, even desperate, side in these talks, even though **Iran** is the country whose economy was sent into a deep chill by international sanctions; the country whose only significant export, oil, lost more than half of its value in recent months. The reason is that **Obama** has a huge political stake in these negotiations. The President may insist that **the United States** will choose no deal over a bad deal, but few people truly believe he has a credible Plan B. Few believe it, particularly in **the Middle East** and notably among **America's Arab** friends, who hold the view that **Iran** is running circles around **the United States** and outplayed **Obama**. As the writer **David Rothkopf** aptly put it, "**Iran** is having a great **Obama** administration." That's a belief that has already started shaking up the region. **Saudi Arabia** has said that it will pursue nuclear weapons if it believes **Iran** has not been stopped, and there is little doubt that other countries among **Iran's Muslim** rivals will do the same. In fact, the notion that **Obama** is not handling the **Iranian** threat effectively is contributing to a new war in **Yemen**, where **Saudi Arabia** and other **Arabs** are trying to push back against gains by **Iran's** allies. We can trace it all back to the **Democratic** primaries in 2007, when then-Sen. **Obama** said he would meet **Iran's** leaders "without preconditions," leading his rival, **Hillary Clinton**, to call the idea "Irresponsible and frankly naive." As the years of his presidency unfolded, and **the Middle East** started coming apart, finding a deal with **Iran** started to look like the one major foreign policy achievement **Obama** might leave behind. The political imperative started to intrude in strategic considerations on an issue that is of transcendent importance to world peace. The framework agreement announced on Thursday came two days after **Obama's** March 31 deadline. The U.S.-imposed deadline served only to pressure **the United States**, and the **French** ambassador very publicly decried as a "bad tactic." That bad tactic was a political move, a push to produce some sort of result, however vague, to protect the talks from critics. Again, a solid agreement that ensures **Iran** will not produce nuclear weapons would be a most welcome development. But the agreement so far does not look promising. It certainly shows the final outcome will differ greatly from what **Obama** had vowed. In a presidential debate in 2012, **Obama** described a crystal clear goal for negotiations. "The deal we'll accept is they end their nuclear program. It's very straightforward." Nobody is talking about **Iran** ending its nuclear program. Not even close. **Iran** will be allowed to keep one-third of its more than 6,000 centrifuges. That's not a small symbolic number. And it does not appear as though any of its nuclear facilities will be dismantled, although **Fordow** will contain no nuclear materials. **Iran** has insisted all along that its nuclear program has only civilian uses. The fact is that **Iran** has a well-established record of lying and concealing the elements of its nuclear program to U.N. inspectors. And the U.N. agency chief says that has not stopped. A couple of weeks ago, with days left until the negotiating deadline, U.N. nuclear chief **Yukiya Amano** said **Iran** is still stonewalling. "We are still not in a position to conclude that all nuclear material in **Iran** is [for a] peaceful purpose," he warned. The negotiations' starting point is that **Iran** would like to have the bomb and the international community wants to delay that as much as possible – and preferably, forever. The world only learned about **Iran's** secret facilities at **Arak and Natanz** after dissidents raised the alarm. **Iran**, we have learned repeatedly, is very good at lying to international inspectors. It is well-established that it has had something to hide about its nuclear program. It is well-established that many of **Iran's** neighbors don't trust it and are anxious about the U.S.-led international dealings with **Iran**. It is well-established that **Iran** has engaged in international terrorism and in destabilizing the region. It is also clear that it took harsh international sanctions and a collapse in oil prices to bring **Iran** to the negotiating table. It was **Iran** that had the most to lose from a failure of talks. But political considerations turned **the United States** into the supplicant. The framework agreement starts lifting those indispensable sanctions much too soon...

Original article (cont.)

...Nuclear enrichment will continue, although at a lower level. **Iran** officially, legally, becomes a nuclear threshold state, with the capability to make the final dash to a bomb within a "breakout" period of one year, the time when presumably inspectors would discover violation and allow the rest of the world to act. Even the **Fordow** facility, conveniently inside a fortified bunker in a mountain, will remain in existence, though "converted" to a nuclear "research facility" And without nuclear material on site. International sanctions lifting will begin almost immediately. Its nuclear infrastructure will remain largely in place, even if operating at a reduced pace, giving **Iran** much of what it wanted. With **Iranian** forces gaining ground in **Arab** lands and **Iranian** commanders declaring the destruction of **Israel** "nonnegotiable" and threatening **Saudi Arabia**, this deal does not look reassuring. **Obama** is right that a diplomatic solution is the most desirable option. But the deal so far looks like (another) win for **Iran**. It introduces enough restrictions that it could give the President the political cover he wants, but it does not do enough to make the world safe from nuclear proliferation and more potentially catastrophic instability in **the Middle East**.

Base T5 summary

sally kohn: if deal collapses, he may fear historians will conclude his legacy collapsed with it. **kohn**: if deal collapses, u.s. will choose no deal over a bad deal, but few believe it. **kohn**: if deal collapses, u.s. will pursue nuclear weapons if it believes **Iran** has not been stopped.

Fine-tuned T5 summary

president **Barack Obama** tied himself to the mast of a nuclear deal with **Iran** even before he became the **Democratic** candidate for president. if the deal collapses, he may fear historians will conclude that his legacy in global affairs collapsed with it. the notion that **Obama** is not handling the **Iranian** threat effectively is contributing to a new war in **Yemen**, where **Saudi Arabia** and other **Arabs** are trying to push back against gains by **Iran**'s allies.

Table 7: Summaries generated using beam search with beam size 5 from the T5-small model and from the one fine-tuned with the objective of producing at least 4 named entities that are in the source document, as described in Section 5.2. Highlighted in purple are the named entities in the text recognized by SpaCy.

Additional responses to “What’s the best piece of advice you’ve ever been given?”	E	S	F	P
<i>before tuning</i>				
My best advice would be to study hard, I overschedule my classes a lot.	0.39	0.35	0.62	0.48
Definitely the best. Knowing how to do something effectively so I don’t have to feel unsure of answers on the road.	0.21	0.38	0.37	0.35
That it is a jump from my everyday life to be completely independent always with my money.	0.48	0.4	0.72	0.63
I am not sure that I have answered all that so far... But I will say, I learned that it is impossible to fail an exam without the right approach.	0.71	0.92	0.83	0.41
I think it would probably be all of the advice I had! Particularly given that I was given the step by asking my best friend for advice, which he then retired from the military to quit smoking and was willing to get me back on my feet!	0.73	0.44	0.74	0.52
Not that I know of but I keep talking about my parents’ divorce symptoms and cutting out custody.	0.63	0.3	0.69	0.53
I just took a job in residence space and got 6 months of paid vacation from it!	0.7	0.3	0.7	0.53
It is some things like going to the dentist or having great insurance. I keep my fingers crossed that they catch a glitch.	0.6	0.14	0.62	0.62
I think the best advice I’ve got is to take things fairly literal. So if something goes wrong and a mistake happens, it’s not enough.	0.67	0.36	0.7	0.58
Mine is staying confident. It’s tough though when I don’t really have advice sometimes, even though I’ve been here 14 years.	0.69	0.33	0.58	0.36
<i>after tuning</i>				
Yes you can never be too careful. Never give up on trying. So what steps does your advice on improving your diet and nutrition?	0.48	0.9	0.7	0.72
I would need to go one with the best military advice. The military lifestyle comes with tough history and I have been expressed with great disputes lol	0.59	0.36	0.45	0.69
Oh, I’d go with my father who taught me to not over educate. He does not care what anyone else thinks. I like to see that people do well.	0.65	0.79	0.77	0.47
Maybe to look inside herself to understand the means of self-employment? It might be too much for me, but it may be helpful. A standard goal is to meet your expectations for success. I wish you well.	0.69	0.67	0.51	0.72
I really like people of higher quality showing their knowledge of their subjects, as much as I may be receiving gifts.	0.63	0.26	0.88	0.88
That’s great! And you are approaching the most respectable answer of all time. The one that may give you the best advice is knowledge!	0.51	0.71	0.91	0.61
When I first came to tell the stories I was working at, the anticipation increased on the meeting before I even started. It felt good to use all my knowledge but then I realized as a customer I could hardly do my own work.	0.77	0.59	0.75	0.64
I think saying, "but really only time, you need to work hard to succeed" and to not give up entirely even when you know you have it.	0.7	0.61	0.77	0.53
I’d have to say knowledge and dedication are definitely what keep me from failing.	0.64	0.76	0.8	0.65
I’ve had so many long years of what some people might call being decent to them. Ever since I found people I knew when I graduated high school I kind of crushed on them and let them know I was still here. Contributing to their faith in life was something I found.	0.87	0.81	0.5	0.74

Table 8: Personality Type ESFP score for BlenderBot’s randomly obtained samples, before and after tuning with the objective of producing responses with 0.8 score on average in each of the ESFP dimensions.

A Hyperparameter Optimization Toolkit for Neural Machine Translation Research

Xuan Zhang and Kevin Duh and Paul McNamee

Johns Hopkins University

Baltimore, Maryland, USA

xuanzhang@jhu.edu; kevinduh@cs.jhu.edu; mcnamee@jhu.edu

Abstract

Hyperparameter optimization is an important but often overlooked process in the research of deep learning technologies. To obtain a good model, one must carefully tune hyperparameters that determine the architecture and training algorithm. Insufficient tuning may result in poor results, while inequitable tuning may lead to exaggerated differences between models. We present a hyperparameter optimization toolkit for neural machine translation (NMT) to help researchers focus their time on the creative rather than the mundane. The toolkit is implemented as a wrapper on top of the open-source Sockeye NMT software. Using the Asynchronous Successive Halving Algorithm (ASHA), we demonstrate that it is possible to discover near-optimal models under a computational budget with little effort.¹

1 Introduction

Deep learning models are difficult to train. Although they achieve impressive results on many tasks, non-trivial amounts of effort are required for selecting appropriate hyperparameters, such as the number of layers, vocabulary size, embedding dimension, and optimization algorithm. This trial-and-error process is necessary for each task, domain, or language. Further, the rapid development of new neural network architectures implies that this hyperparameter optimization process will only become more expensive.

Currently, hyperparameter optimization tends to be performed manually by researchers in an ad hoc fashion, using scripts put together independently. The lack of open-source support tools means that the level of rigor in hyperparameter optimization may vary widely. This poses two risks:

1. **Insufficient exploration** of the hyperparameter space may lead to poor results, killing an otherwise promising research idea.
2. **Inequitable allocation** of compute resources for hyperparameter optimization of one model over another may lead to exaggerated results differences and misleading conclusions.

The importance of documenting the hyperparameter optimization process in research has already been widely recognized and is included as an item under the “Responsible NLP Checklist”² required for paper submissions in the field. To support these efforts, we believe it will be beneficial to develop open-source tools to improve the hyperparameter optimization process itself.

This paper presents a *hyperparameter optimization toolkit* for NMT research. It enables researchers to easily explore the hyperparameter space of various NMT models based on the PyTorch codebase of AWS Sockeye framework (Hieber et al., 2022). One simply specifies (1) the desired set of hyperparameter options to search, (2) the compute resource constraints, and (3) the training data paths, then the toolkit will plan and execute an automatic hyperparameter optimization and return the best model discovered. The toolkit implements the Asynchronous Successive Halving Algorithm (ASHA) (Li et al., 2020), which is well-suited for commodity off-the-shelf distributed grids.

In the following, we first give an overview of the toolkit (Section 2) and hyperparameter optimization algorithm (Section 3). Then, the case study in Section 4 illustrates how the toolkit can help a researcher search over thousands of hyperparameter configurations with ease. Finally, Section 5 discusses our design choices, hopefully serving as a

¹<https://github.com/kevinduh/sockeye-recipes3> (code), <https://cs.jhu.edu/~kevinduh/j/demo.mp4> (video demo)

²<https://aclrollingreview.org/responsibleNLPresearch/>

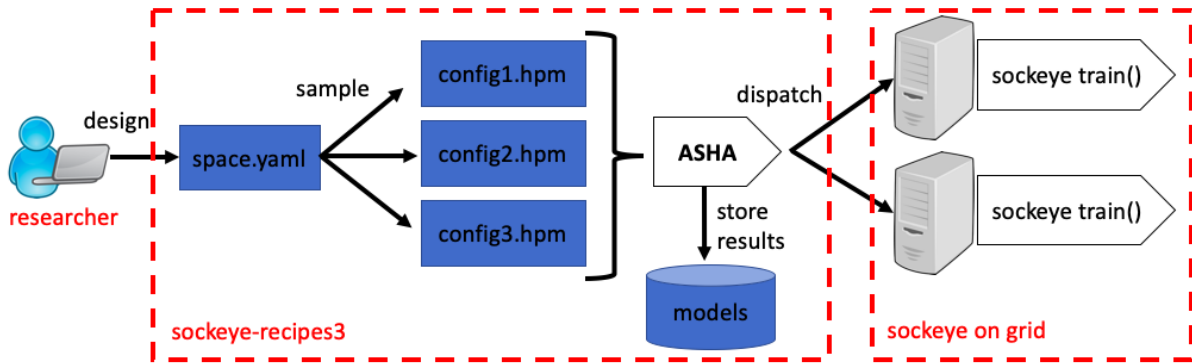


Figure 1: An overview of the sockeye-recipes3 hyperparameter optimization toolkit

reference for those who want to implement similar toolkits for different NLP software.

2 Usage Overview

Our hyperparameter optimization toolkit is named sockeye-recipes3, since it cooks up different models by training models with the AWS Sockeye NMT framework, version 3. An overview is shown in Figure 1. For concreteness, let us suppose the researcher in Figure 1 wants to run a rigorous hyperparameter optimization to obtain a strong Transformer baseline for a new dataset.

Step 1: The researcher designs a hyperparameter search space for his/her model. Table 1 shows some common hyperparameters for Transformers, but the toolkit is flexible to incorporate any user-defined hyperparameter. This hyperparameter space is expressed as a YAML file, e.g. space.yaml:

```
transformer_model_size: [256, 512, 1024]
transformer_attention_heads: 8
transformer_feed_forward_num_hidden: [1024, 2048]
...
```

The snippet above indicates that the researcher wishes to explore three choices for model size, one choice for attention head, and two choices for a feed-forward number of hidden units. The Cartesian product of all these choices forms the full hyperparameter space.

Step 2: sockeye-recipes3 samples from the full hyperparameter space to generate a set of bash files called hpm files. Each hpm file represents a *specific* hyperparameter configuration and encapsulates all the information needed to train a model. This includes not only hyperparameter settings but also paths to training and validation data. For example, config1.hpm might train a model with:

```
transformer_model_size=256
```

```
transformer_attention_heads=8
transformer_feed_forward_num_hidden=1024
train_data=~ /data/wmt.train.de-en.bitext
validation_data=~ /data/wmt.dev.de-en.bitext
```

The set of hpm files represents all the hyperparameter configurations to be explored by the hyperparameter optimization algorithm. Rather than randomly sampling a subspace, one can also generate the full Cartesian product or manually edit some hpm files based on prior knowledge. Depending on the researcher’s usage scenario, this set typically numbers from tens to thousands.

Step 3: Once the researcher is ready, he/she starts the ASHA program with resource specifications such as the number of concurrent GPUs to use and the number of checkpoints per training run. This Python code dispatches the training processes as standard Sockeye jobs to a distributed grid.³ ASHA will attempt to efficiently train as many models as possible given the computational constraints. It is a bandit learning method that automatically learns when to stop a not-so-promising training run in order to allocate resources to other hyperparameter configurations. Details are in Section 3.

Step 4: The results of all Sockeye training runs dispatched by ASHA are stored on disk. Each hpm file will have a corresponding subdirectory with the output log of a Sockeye training process. This makes it easy to replicate or continue any training runs in the future, with or without the sockeye-recipes3 toolkit. Ultimately, the researcher can pick out the best model from the set for further experimentation.

³The dispatch in sockeye-recipes3 is currently implemented for the Univa Grid Engine (UGE) but is easily extendable to other similar grid management software like SLURM.

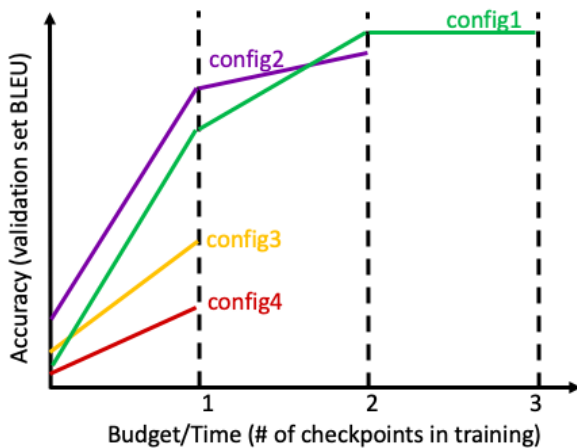


Figure 2: Illustration of Successive Halving

Additional features: (a) Metric: The toolkit’s default is to find models with high BLEU on the validation set. This can be changed to any user-specified metric. Also, we have devised a multi-objective version of ASHA to enable joint optimization of accuracy and inference speed based on Pareto optimality (Marler and Arora, 2004).

(b) Analysis: After an ASHA run, one may wish to see if there are certain trends in hyperparameters, e.g. are some more important than others? This introspection can be helpful in understanding the model or designing future hyperparameter spaces. We have included a tool for posthoc analysis using Explainable Boosting Machines (Deb et al., 2022).

3 Hyperparameter Opt. with ASHA

Problem: Suppose we have N hyperparameter configurations (hpm files) and a max compute budget of B , measured in terms of the total number of training checkpoints available. Let us select n configurations for actual training, where $n \leq N$. If each configuration is allocated the same budget, then each would be trained up to B/n checkpoints. When N is large, we have an untenable problem:

- If we choose n to be large (close to N), then B/n will be small, indicating that each configuration is only trained for a few checkpoints. Most models likely will not have converged.
- If we choose n to be small (despite N being large), then configurations that are chosen are trained well (large B/n) but the majority of configurations are not even trained at all.

The only solution is to allocate each configuration with a variable budget: i.e. train promising

configurations for more checkpoints and terminate the not-so-promising ones prior to convergence. This is an intuitive idea that has probably been performed countless times by researchers by tracking learning curves in a manual fashion.

Successive Halving: The Successive Halving Algorithm (Jamieson and Talwalkar, 2016) implements this intuition algorithmically, and is illustrated in Figure 2. Suppose we choose $n = 4$ hyperparameter configurations to explore and the total budget is $B = 7$ checkpoints. We begin by first training each configuration up to checkpoint 1 and measuring their validation accuracy. The configurations with lower accuracies at this point (config3, config4) are deemed not-so-promising and are terminated. The remaining half (config1, config2) are trained longer, and validation accuracy is measured again at checkpoint 2. Again, half of the configurations are terminated and the other half is “promoted” to be trained longer; this is done successively until the total budget is reached.

The main assumption of Successive Halving is that learning curves of different configurations are comparable and that the relative ranking of validation accuracy at intermediate checkpoints correlates to that at convergence. This is an assumption that cannot be proved but is likely reasonable in most cases with the proper setting of checkpoint intervals.

ASHA: In practice, the Successive Halving Algorithm as described above has a bottleneck at each checkpoint: we need to wait for all configurations to return their validation score before deciding the best half to promote. The actual time that a configuration needs to reach a checkpoint depends on many factors such as GPU device type and model size. So we may end up waiting for the slowest training run, causing poor grid utilization.

To address this, an Asynchronous Successive Halving Algorithm (ASHA) is introduced (Li et al., 2020). The idea is to promote a configuration as soon as it is guaranteed to be in the top half, without waiting for all configurations to return with their checkpoints’ validation accuracy. For example in Figure 2, suppose three configurations (e.g. config2, config3, config4) have already returned an accuracy for checkpoint 1. We are then safe to promote the best one out of the group (config2) without waiting for config1 to return since config2 will be among the top half regardless of config1’s accuracy.

Name & Description	Settings
Architecture Hyperparameters	
transformer_model_size - size of model/embeddings	{256, 512, 1024}
transformer_attention_heads - # of heads	8
transformer_feed_forward_num_hidden - # units in feedforward layer	{1024, 2048}
num_layers - for "encoder:decoder"	{6:6, 8:4, 4:4, 6:2}
Data Pre-processing Hyperparameters	
bpe_symbols_src - # of BPE symbols on source side	{5k, 10k, 30k}
bpe_symbols_trg - # of BPE symbols on target side	{5k, 10k, 30k}
Training Hyperparameters	
optimized_metric	perplexity
initial_learning_rate: initial rate for ADAM optimizer	{0.0002, 0.001, 0.002}
embed_dropout - dropout rate for source:target embeddings	.0:.0
label_smoothing	0.1
seed - random initialization seed	{1, 2}
Hardware-related Hyperparameters	
batch_size - # of words in batch	4096
checkpoint_interval - #batches before saving checkpoint to disk	4000

Table 1: Hyperparameter space used in the case study. The settings in red font are searched over, while others are held fixed. In total, we will explore $3 \times 2 \times 4 \times 3 \times 3 \times 3 \times 2 = 1296$ configurations.

Please refer to the original papers on ASHA, Successive Halving, and a variant called Hyperband (Li et al., 2016) for more detailed analyses. We focus on ASHA in sockeye-recipes3.

4 Case Study

Goal: To illustrate how sockeye-recipes3 works in practice, we show a case study on building a strong Transformer baseline for a new Telugu-to-English dataset. Our initial training set consists of 900k lines of bitext obtained from public sources via the OPUS portal (Tiedemann, 2012). This is augmented with 7 million lines of back-translated data obtained by running a reverse system (English-to-Telugu NMT trained on 900k) on web-scraped news from the Leipzig corpus (Goldhahn et al., 2012). 3000 lines are held out from the initial training set to serve as the validation set.

Given this setup, our goal is to run hyperparameter optimization on a standard Transformer architecture to obtain the best possible model according to validation BLEU. This model can serve as a strong baseline for any future NMT experiment based on the same dataset. Since this is a low-resource language pair that is relatively unexplored in the research community, we opt to search a large hyperparameter space.

Hyperparameter space: Our space.yaml file is defined according to the options listed in Table 1. While any user-defined hyperparameter is possible, sockeye-recipes3 exposes the most common options. We explore a total of 1296 configurations.

ASHA run: We run ASHA using the resource settings in Table 2. The reduction rate decides the fraction of configurations that are promoted each time: a factor $p=2$ reduction rate corresponds to "halving", but in practice, one can choose to be more or less aggressive. We also specify the number of GPUs that can be used concurrently by ASHA: here, it will dispatch jobs asynchronously up to that limit of $G=40$.

Finally, the settings for a min, max, and per-rung checkpoints are NMT-specific modifications we found useful for ASHA. In Figure 2, halving is performed at each checkpoint, or at each "rung" in ASHA terminology. It is convenient to give NMT researchers the flexibility to choose the exact schedule: here, we decide that each configuration is trained for at least $r=5$ checkpoints (corresponding to 5×4000 batches due to the checkpoint_interval in Table 1) before we perform successive halving at the first rung. Thereafter, each configuration is trained for $u=2$ checkpoints before successive halving is performed. Finally, no configurations will be trained with more than $R=25$

Reduction rate. Top 1/p promoted	p=2
# of GPUs available	G=40
min checkpoints per model	r=5
#checkpoints per config per rung	u=2
max checkpoints per model	R=25

Table 2: ASHA settings for case study

checkpoints regardless of other ASHA settings; this small number of maximum checkpoints will probably not obtain state-of-the-art results but is suitable for the purpose of discovering several good configurations. The researcher may first inspect the ASHA results to identify several promising configurations, then manually train them for longer.⁴

Figure 3 samples a few learning curves (out of the 1296 configurations in total) to demonstrate how ASHA works in practice. The top figure is analogous to Successive Halving in Figure 2, while the bottom figure shows how the asynchronous dispatch occurs over time.

Comparison with grid search: To confirm whether ASHA finds good models, we also run a grid search on the same 1296 configurations, training each with up to 25 checkpoints. This corresponds to a total cost of $25 \times 1296 = 32,400$. In comparison, the ASHA run in our case study costs 60% less at 9066 checkpoints in total.

Table 3 confirms that ASHA can find good models that are found by an exhaustive grid search. For example, the maximum BLEU score by grid search is 20.3, and while this model is terminated at rung 4, the final model discovered by ASHA has a competitive BLEU score of 20.1. In our experience, ASHA is effective at finding a set of reasonable models at a fraction of the computational cost; if we desire the best possible model, nothing can replace the manual effort of an experienced researcher.

5 Design

sockeye-recipes3 is designed with two principles: (1) All NMT codes, such as a researcher’s proposed extension of the Sockeye framework, are encapsulated in separate conda environments. (2) All hyperparameters and data paths (for baseline and proposed methods) are explicitly specified in hpm files, and stored together with each sockeye

⁴The best model discovered has 8 encoder layers, 4 decoder layers, 1024 model size, 2048 feedforward size, 10k source subwords, 30k target subwords, and achieves 35.6 spBLEU on the FLORES101 devtest (Goyal et al., 2022).

rung	ckpt	config	budget	med	max
0	5	1296	6480	0.3	20.3
1	7	648	7776	17.2	20.3
2	9	324	8424	18.9	20.3
3	11	162	8748	19.4	20.3
4	13	81	8910	19.7	20.3
5	15	40	8990	19.7	20.1
6	17	20	9030	19.7	20.1
7	19	10	9050	19.8	20.1
8	21	5	9060	19.8	20.1
9	23	2	9064	20.0	20.1
10	25	1	9066	20.1	20.1

Table 3: ASHA vs. Grid search: Each row lists the # of configurations explored in each rung, # of checkpoints (ckpt) trained so far per configuration, and accumulated budget (total checkpoints). The med/max columns are median/max BLEU scores among the configurations explored if they were trained to completion in a grid search. For example, in rung 2, 324 configurations were explored by ASHA and trained up to 9 checkpoints. If they were trained up to the full 25 checkpoints and their BLEU scores were collected, the median would be 18.9 and the max would be 20.3. ASHA preserves many of the top configurations that would be found by grid search.

training run. This means that it is easy to replicate or continue any training run by referring to (1) and (2). ASHA dispatches will run Sockeye training for u checkpoints at a time, so a job will automatically return the GPU resource at the end of each rung.

The ASHA implementation is a Python script that sits on a single server and regularly checks the status of Sockeye training runs on the distributed grid setup. The pseudocode is shown in Algorithm 1. The script keeps track of configurations that are training or paused at a checkpoint. When there is an idle GPU, it will decide whether to explore a new hpm or promote an existing one. The dispatch is a job submission command that starts a Sockeye train process on a GPU node. It depends only on the conda-environment provided, so it is easy to optimize different NMT implementations by exchanging the environment while keeping similar space.yaml, leading to equitable tuning.

6 Related Work

ASHA and variants can be viewed as bandit algorithms that balance exploration (trying new configurations) with exploitation (training the current configurations for longer). They obtain efficiency

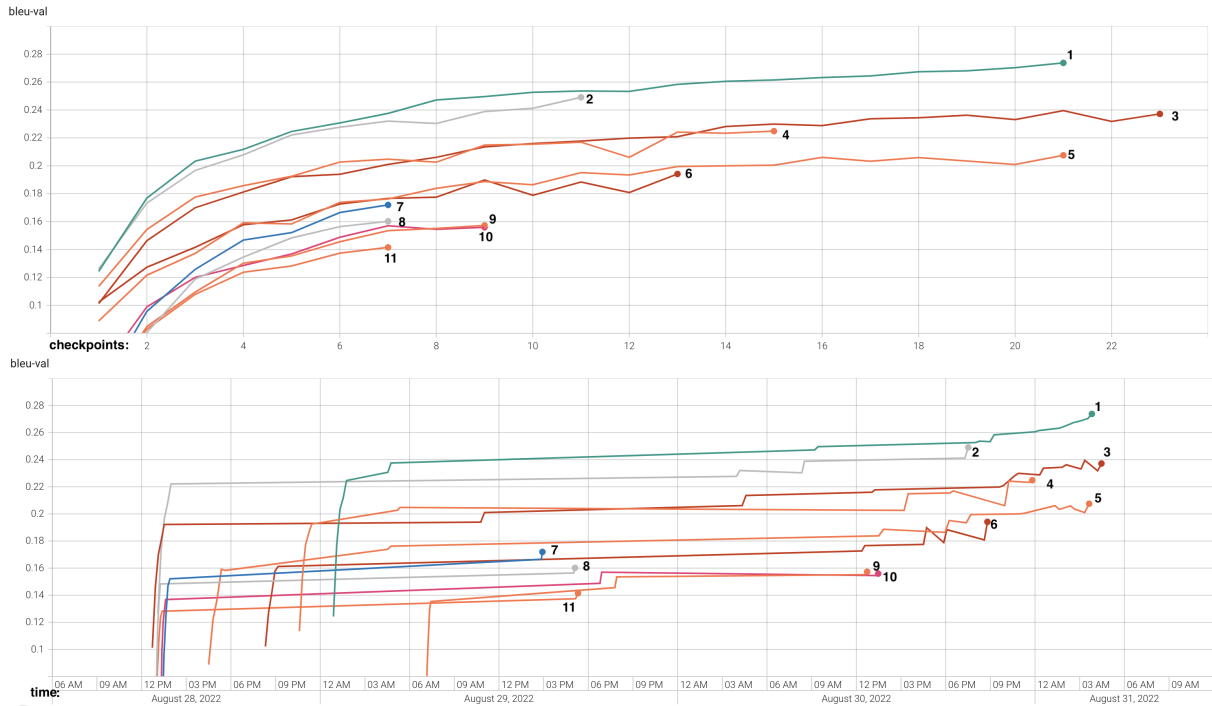


Figure 3: Learning curves for a random sample of configurations in ASHA. The y-axis is the validation BLEU score. The top figure, where the x-axis represents # of checkpoints, is analogous to Figure 2 and shows which configurations are promoted. The bottom figure represents the same configurations plotted against wallclock time on the x-axis; this illustrates the asynchronous nature of ASHA. Observe that configurations are not started in sync, and long plateaus indicate when ASHA decided to pause the configuration at a checkpoint to allocate GPUs for other ones.

Algorithm 1 ASHA pseudocode

```

while budget remains do
  for all  $c \in \text{configs}$  do
     $s = \text{check\_state}(c)$   $\triangleright$  Still training or at checkpoint?
  end for
  for all  $g \in \text{idle GPU}$  do
     $h = \text{get\_hpm}(\text{configs})$   $\triangleright$  Explore new or promote?
     $\text{dispatch}(h, g, \text{conda-env})$   $\triangleright$  Sockeye train()
  end for
  pause for  $m$  minutes
end while

```

by early stopping. Another class of methods are “blackbox” optimizers, e.g. Bayesian Optimization and Evolutionary Methods (Feurer and Hutter, 2019): they treat hyperparameters as input features, observed accuracy as output targets, and train a proxy model to predict new hyperparameters that are worth sampling. These two classes of methods can be combined (Falkner et al., 2018); this is potentially future work. Several benchmarks provide comparisons of state-of-the-art (Zhang and Duh, 2020; Zöllner and Huber, 2021).

Neural Architecture Search (Elsken et al., 2019) is related to hyperparameter optimization but focuses more on fine-grained choices (e.g. changing skip connections at different layers). This is an active area of research, but out-of-scope for our purpose of improving NMT experimentation.

There are some existing toolkits like Vizier (Song et al., 2022) and Ray Tune (Liaw et al., 2018), which are suitable for those wanting general rather than application-specific solutions.

7 Conclusions

There is a progression of toolkit development that enables researchers to do better work. Deep learning toolkits like PyTorch and Tensorflow made it easy to exploit GPU hardware. Application-specific toolkits like Sockeye and Fairseq build on top of that, and enabled researchers to quickly prototype new ideas. Further on top, we believe that hyperparameter optimization toolkits and experiment management toolkits in general will further help advance the speed and rigor of research.

We presented sockeye-recipes3, an open-

source hyperparameter optimization toolkit for NMT research. Our hope is this will relieve some of the mundane aspects of manual hyperparameter tuning so that researchers can focus on more creative activities. A rigorous and automated hyperparameter optimization process will also lead to more trustworthy experiment results.

Limitations

Scope of support: The `sockeye-recipes3` toolkit only supports the AWS Sockeye NMT framework. It is suitable for researchers who plan to implement and test out different NMT models in PyTorch using Sockeye’s codebase. It is not meant to be extensible to hyperparameter optimization methods for other frameworks in NLP. The reason is that each toolkit has its own nuanced error messages and hyperparameter definitions, so it is easier to do design a focused toolkit.

No guarantees: In general, hyperparameter optimization methods give no theoretical guarantees; there is always an aspect of uncertainty. For example, there is no guarantee that ASHA will keep the top configurations if the learning curves do not follow our assumptions. One may be more conservative by setting more checkpoints per rung in ASHA, but this decreases the potential for efficiency.

Manual design: `sockeye-recipes3` does not fully automate the entire model-building process. The researcher still needs to design the hyperparameter space for each task. This search space is critical for the success of ASHA that follows. One may imagine a transfer learning (or meta-learning) approach where hyperparameter spaces from similar tasks are borrowed, but this is currently an open problem.

Ethics Statement

Automated hyperparameter optimization can lead to efficiencies in model building, but we need to be cognizant that there is also a risk of excessive optimization. The user needs to design what is a reasonable search space: for example, would it be worthwhile to optimize over many different random initialization seeds or over small differences between model sizes?

Excessive optimization poses three risks: First, one may select models that “overfit”, though this can be ameliorated by proper choices of validation sets. Second, hyperparameter optimization gives an

advantage to research teams with large compute resources; ASHA and similar methods are not useful on grids with less than e.g. 10 GPUs.

Third and perhaps more important, the computation may be wasteful. “Green AI” is an important call-to-arms for the research community: hyperparameter optimization is a double-edged sword in that proper usage leads to efficiency while excessive usage leads to wastefulness.

For example, to quantify the CO₂e emissions in our case study, we estimate that ASHA and grid search spent a total of 3050 hours on GPU compute node. Our grid contains a mix of NVIDIA TITAN RTX, GeForce RTX 2080 Ti, and Tesla V100. In future versions of `sockeye-recipes3`, we plan to track power use individually for all jobs but let us assume an average power consumption of 250 watts, for a total of 0.762MWh. If we assume carbon efficiency⁵ is at 432 kg CO₂e per MWh, data center power usage effectiveness (PUE) is 1.5, and there are no additional offsets for renewable energy, we end up with:

$$\frac{0.762 \text{ MWh}}{1} \times \frac{432 \text{ kg}}{\text{MWh}} \times \frac{1.5}{1} = 494 \text{ kg CO}_2\text{e} \quad (1)$$

This corresponds to the CO₂e of driving a car for 2000km or burning 247kg of coal. Ideally, we will eventually reach an understanding as a community of what amount of use is appropriate or excessive.

References

- Kiron Deb, Xuan Zhang, and Kevin Duh. 2022. [Post-hoc interpretation of transformer hyperparameters with explainable boosting machines](#). In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 51–61, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. [Neural architecture search: A survey](#). *Journal of Machine Learning Research*, 20(55):1–21.
- Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. [Bohb: Robust and efficient hyperparameter optimization at scale](#). In *International Conference on Machine Learning*.
- Matthias Feurer and Frank Hutter. 2019. [Hyperparameter optimization](#). In *Automated Machine Learning*, pages 3–33. Springer.
- Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. [Building large monolingual dictionaries at the](#)

⁵<https://mlco2.github.io/impact/>

- Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).
- Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc’ Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. [The Flores-101 evaluation benchmark for low-resource and multilingual machine translation](#). *Transactions of the Association for Computational Linguistics*, 10:522–538.
- Felix Hieber, Michael Denkowski, Tobias Domhan, Barbara Darques Barros, Celina Dong Ye, Xing Niu, Cuong Hoang, Ke Tran, Benjamin Hsu, Maria Nadejde, Surafel Lakew, Prashant Mathur, Anna Currey, and Marcello Federico. 2022. [Sockeye 3: Fast neural machine translation with pytorch](#).
- Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR.
- Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. [A system for massively parallel hyperparameter tuning](#). In *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246.
- Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet S. Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18:185:1–185:52.
- Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.
- R. Marler and Jasbir Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.
- Xingyou Song, Sagi Perel, Chan Lee Lee, Greg Kochanski, and Daniel Golovin. 2022. Open source vizier: Distributed infrastructure and api for reliable and flexible blackbox optimization. *ArXiv*, abs/2207.13676.
- Jörg Tiedemann. 2012. [Parallel data, tools and interfaces in OPUS](#). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).
- Xuan Zhang and Kevin Duh. 2020. [Reproducible and efficient benchmarks for hyperparameter optimization of neural machine translation systems](#). *Transactions of the Association for Computational Linguistics*, 8:393–408.
- Marc-André Zöllner and Marco F. Huber. 2021. [Benchmark and survey of automated machine learning frameworks](#). *J. Artif. Int. Res.*, 70:409–472.

Japanese-to-English Simultaneous Dubbing Prototype

Xiaolin Wang and Masao Utiyama and Eiichiro Sumita

Advanced Translation Research and Development Promotion Center
National Institute of Information and Communications Technology, Japan
{xiaolin.wang,mutiyama,eiichiro.sumita}@nict.go.jp

Abstract

Live video streaming has become an important form of communication such as virtual conferences. However, for cross-language communication in live video streaming, reading subtitles degrades the viewing experience. To address this problem, our simultaneous dubbing prototype translates and replaces the original speech of a live video stream in a simultaneous manner. Tests on a collection of 90 public videos show that our system achieves a low average latency of 11.90 seconds for smooth playback. Our method is general and can be extended to other language pairs.

1 Introduction

Live video streaming over the Internet has become a very important form of communication in human society. It has many advantages such as fast, not constrained by distance, economical and safe.

If the language barrier (Ahmad Abuarqoub, 2019) can be broken down in live video streaming, it will greatly promote global communication. However, the current common solution to cross-language live video streaming is to use automatic simultaneous interpretation (Müller et al., 2016; Wang et al., 2016; Franceschini et al., 2020; Bojar et al., 2021) to display translated subtitles. Reading subtitles at the bottom of the screen is uncomfortable and degrades the viewing experience (Wissmath et al., 2009).

Our simultaneous dubbing prototype aims to help live video streaming break down language barriers. Our prototype translates and replaces the original speech of a live video stream, creating a seamless viewing experience in the target language. Table 1 summarizes what our system is. Our system consists of a complete simultaneous interpretation system and a simplified automatic language dubbing system (Furukawa et al., 2016; Yang et al., 2020; Öktem et al., 2019; Federico et al., 2020). By

Feature	SI	LD	Ours
Speech Recognition	✓	✓	✓
Machine Translation	✓	✓	✓
Low Latency	✓		✓
Text-to-Speech		✓	✓
Duration Match		✓	✓
Audio Rendering		✓	
Lip Sync		✓	
Live Streaming			✓

Table 1: Comparison of automatic simultaneous interpretation (SI), automatic language dubbing (LD) and our system.

combining these two technologies, it gains a novel ability of live video streaming in a target language.

Tests on a collection of 90 public videos show that the live streaming from our system achieves a low average latency of 11.90 seconds and meets a smoothness criterion. Therefore, our system can be widely used in fields such as news broadcasting, conferences and education. Furthermore, our method is general and can extend to other language pairs.

The main contributions of our work include,

- implementing a first simultaneous dubbing prototype for multi-language live video streaming;
- developing evaluation metrics for the latency, smoothness and duration matching of simultaneous dubbing;
- proposing an adaptive playback method to balance latency and smoothness.

The rest of this paper is organized as follows. First, Section 2 reviews related works. Then, Section 3 describes our method for implementing simultaneous dubbing. After that, Section 4 tests our system on a collection of 90 public videos in

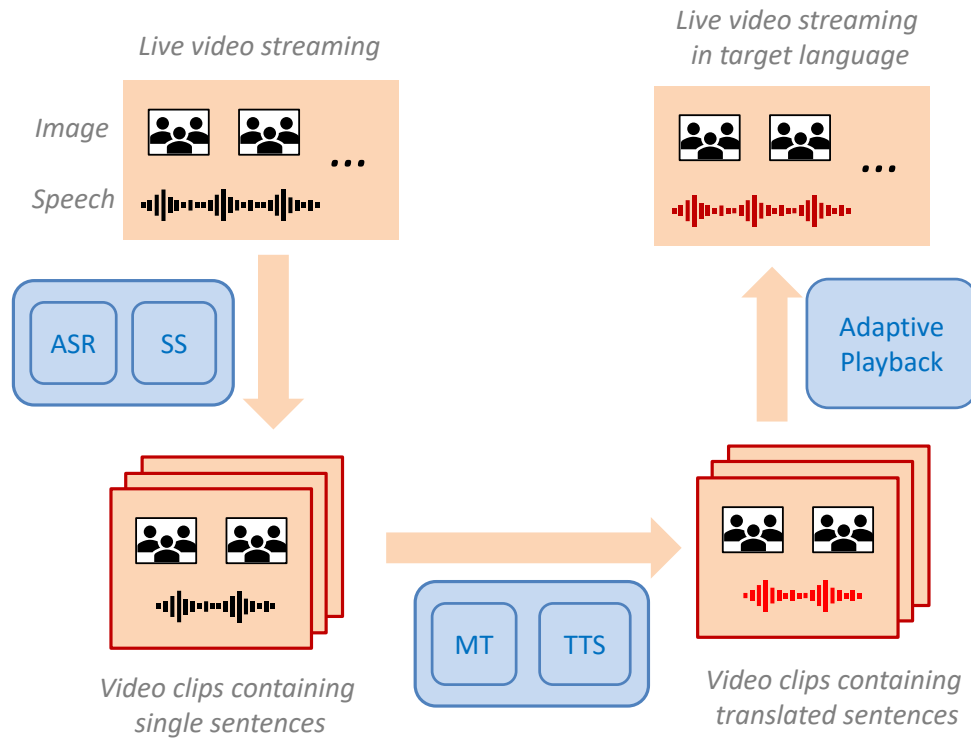


Figure 1: Implementation of simultaneous dubbing using automatic speech recognition (ASR), sentence segmentation (SS), machine translation (MT), text-to-speech (TTS) and adaptive playback.

terms of latency, smoothness and duration matching. Finally, Section 5 concludes this paper with a description on future works.

2 Related Works

Automatic simultaneous interpretation and automatic language dubbing are the two topics most closely related to our work.

2.1 Automatic Simultaneous Interpretation

Simultaneous interpretation is a hot topic. Due to space limitations, we only review some selected practical systems.

Professor Alex Waibel from the Karlsruhe Institute of Technology (KIT) demonstrates a simultaneous interpretation system that automatically translates lectures from German to English in 2012 (Figure 2a)¹. The transcripts are shown on the left part of the window and the translation is shown below.

Microsoft Meetings pilots live translated subtitles in 2022 (Figure 2b)². With this new feature,

¹<https://www.youtube.com/watch?v=GHeHiPh3u0s>

²<https://techcommunity.microsoft.com/t5/microsoft-teams-public-preview/now-in-public-preview-live-translated-captions-in-meetings/mp/3620055>

users can select a translation language for live subtitles. This feature helps users fully participate in meetings where the spoken language may not be their most comfortable language to use. Google Meet has a similar feature³.

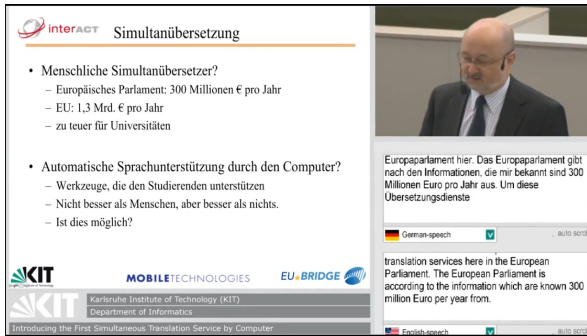
Wang et al. (2022) demonstrate a multimodal simultaneous interpretation system that annotates translation with speakers (Figure 2c). Due to the delays in the process of simultaneous interpretation, it is sometimes difficult for users to trace the translation back to speakers. Thus, the system explicitly presents “who said what” to users.

Our work differs from these related works by presenting translation as dubbing, whereas related works present translation as subtitles. We believe our method can be incorporated into these related works to bring better services to users.

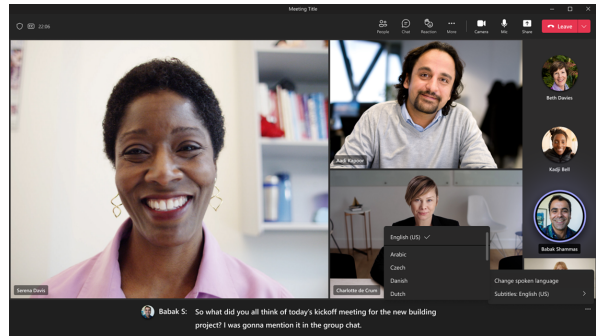
2.2 Automatic Language Dubbing

Automatic Language Dubbing commonly operates on entire video (Yang et al., 2020; Öktem et al., 2019; Federico et al., 2020) whereas our work operates on video streams and generates output in low latency. In addition, due to the complexity of the task, manually correction and adjustment are

³<https://workspaceupdates.googleblog.com/2022/01/live-translated-captions-in-google-meet-generally-available.html>



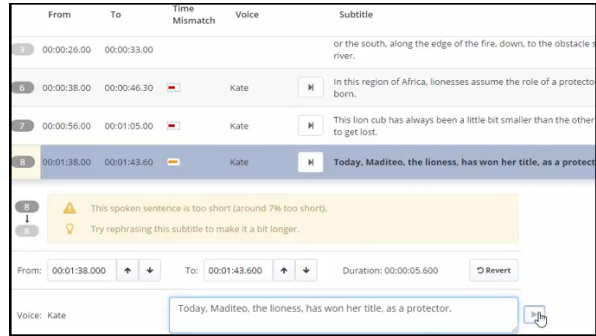
(a) KIT SI of lectures



(b) Microsoft SI of meetings



(c) SI with multimodal speaker recognition



(d) VideoDubber LD

Figure 2: Automatic simultaneous interpretation (SI) and language dubbing (LD) systems.

often required, such as VideoDubber (Figure 2d)⁴, whereas our work is fully automatic.

3 Methods

Our prototype accomplishes simultaneous dubbing through three main steps as (Figure 1),

1. Segmenting the source video stream into video clips that contain one single sentence using automatic speech recognition (Hinton et al., 2012; Graves and Jaitly, 2014) and sentence segmentation (Sridhar et al., 2013; Iranzo-Sánchez et al., 2020). For automatic speech recognition, we use the Transformer-based (Vaswani et al., 2017) acoustic model and the seq2seq criterion (Sutskever et al., 2014; Synnaeve et al., 2019) implemented in Flashlight (Pratap et al., 2019)⁵. For sentence segmentation, we replace the backbone network of CytonNSS (Wang et al., 2019)⁶ with Transformer to improve accuracy.
2. Generating a translated speech waveform for each sentence using machine translation (Bah-

danau et al., 2014; Stahlberg, 2020) and text-to-speech (Wang et al., 2017; Ren et al., 2019). For machine translation, we use the Transformer model implemented in OpenNMT (Klein et al., 2017)⁷. For text-to-speech, we modify the official implementation of VITS (Kim et al., 2021)⁸ to generate speech waveforms from speaker embeddings to match the original voice, similar to (Jia et al., 2018).

3. Playing the images and the translated speech waveforms using an adaptive playback method.

The main challenge of simultaneous dubbing is that the output of sentence segmentation (Step 1) and machine translation (Step 2) is irregular in time, but video streaming is constantly consuming data. For example, in the source stream, someone speaks a sentence for about 15 seconds. The system then spends another 5 seconds generating the translated speech waveform. This results in a 20-second data gap in the output stream.

The adaptive playback method addresses this challenge while maintaining low latency (Figure 3).

⁴https://app.videodubber.com/?source=hp_dub_it_now

⁵<https://github.com/flashlight/flashlight/tree/main/flashlight/app/asr>

⁶<https://github.com/arthurx1w/cytonNss>

⁷<https://github.com/OpenNMT/OpenNMT-py>

⁸<https://github.com/jaywalnut310/vits>

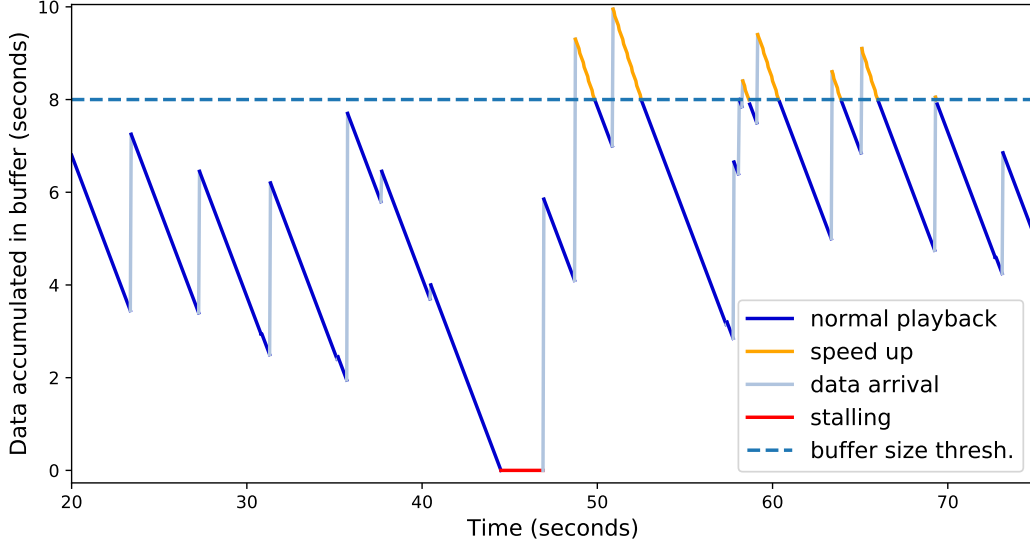


Figure 3: Behaviors of adaptive playback method.

The speed of the playback changes according to the size of the data accumulated in the playback buffer, formulated as,

$$\text{speed} = \begin{cases} 1.0 & \text{if } x < \theta, \\ \alpha & \text{if } x \geq \theta, \end{cases} \quad (1)$$

where x is the amount of data accumulated in the playback buffer. The playback acceleration $\alpha \geq 1$ and the buffer size threshold θ are parameters that control latency and smoothness (Section 4.2).

4 Evaluation

Our system is tested on a collection of 90 public videos of Japanese interviews, speeches, presentations and lectures. The total running time of the collection is approximately 21 hours 45 minutes. The tests are run on a desktop computer equipped with one Intel Xeon E5-2630 V3 CPU and two Nvidia Quadro RTX 4000 GPUs.

The test results are shown in Table 2. The performance of our system is evaluated in terms of latency (Section 4.1), smoothness (Section 4.2) and duration matching (Section 4.3).

Our system presets three modes, Fast, Balance and Quality, for different trade-offs of speed and quality. Users can select the mode according to the application. Table 3 lists the parameters for each mode. Table 7 shows grid search for the buffer size threshold and playback acceleration for the fast mode.

4.1 Latency

Latency is the delay between the input video stream and the output video stream. It is calculated by

comparing the start time of each source sentence in the input stream with that of the corresponding translation, formulated as,

$$\text{Latency} = \frac{\sum_{i=1}^{N_{\text{sent}}} T_{i,s} - T_{i,o}}{N_{\text{sent}}}, \quad (2)$$

where N_{sent} is number of the sentences, $T_{i,o}$ and $T_{i,s}$ are the start times of original waveform and synthesized translated waveform, respectively. Table 5 gives an example with a latency of 9.8.

The fast mode on our system achieves an average latency of 11.90 seconds (Table 2). This is relatively fast as the maximum duration of sentences in each video averages 10.76 seconds and the maximum delay of the generated translated speech averages 15.59 seconds on the whole dataset (Table 4). It is difficult to reduce the latency too much below this value while maintaining smooth video streaming.

4.2 Smoothness

The smoothness of the output stream is measured by,

- **# Stall** : the average number of stalls per minute.
- **S. Dur.** : the total duration of stalls per minute.

This follows the researches on assessing the quality of Internet video streaming (Pastrana-Vidal et al., 2004; Qi and Dai, 2006; Moorthy et al., 2012; Seufert et al., 2014; Garcia et al., 2014; Bampis et al., 2017; Zhou et al., 2022)

Mode	Latency (s) [↓]	Smoothness		Duration Match		
		# Stall. [↓]	S. Dur.(s) [↓]	Fit (%) [↑]	D. Fit (%) [↑]	D. Ex.(%) [↓]
Fast	11.90	1.21	2.55	89.32	71.43	154.03
Balance	12.90	0.71	1.71	90.60	75.50	146.67
Quality	14.12	0.49	1.38	91.50	78.43	126.16

Table 2: Evaluation Results.[↓] the smaller the better. [↑] the higher the better. (s) seconds.

Mode	Playback		MT
	Buf.(s)	Acc.	# Models
Fast	5.0	x 1.06	1
Balance	7.0	x 1.04	2
Quality	9.0	x 1.02	3

Table 3: Paramteres

Video	Max Dur.(s)	Max Delay(s)
1	13.45	15.05
2	10.66	16.80
3	9.97	16.40
4	11.81	17.40
	...	
87	9.09	14.20
88	8.88	14.45
89	8.81	13.00
90	10.86	18.05
Average	10.76	15.59

Table 4: Maximum duration and processing delay per sentence for each video stream using one machine translation model.

Users tend to tolerate up to three short one-second stalls, or one long three-second stall according to the crowdsourcing-based studies (Höbfeld et al., 2011). The fast mode of our system is slightly better than this guideline, while the balance mode and the quality mode are well above this guideline (Table 2).

The smoothness of the streaming is influenced by the buffer size threshold and the acceleration in the adaptive playback module. We perform grid search for these two parameters for the fast mode, balance and quality mode, respectively. Table 7 shows the search result for the fast mode. To speed up the search, we record the ready time of each sentence and simulate on the playback module.

4.3 Duration Matching

Language dubbing requires that the duration of each translated speech waveform matches the duration of its source sentence. The duration matching

is measured as,

- **Fit (%)** : the percentage of the translated speech waveforms that **fit** in their original durations, formulated as,

$$\frac{N_{\text{Fit}}}{N_{\text{Fit}} + N_{\text{Exceed}}} \times 100\%, \quad (3)$$

where N_{Fit} and N_{Exceed} is the number of translated speech waveforms that fit and exceed the original durations, respectively.

- **D. Fit (%)** : the average percentage of the **durations** for the translated waveforms that **fit** the original durations, formulated as,

$$\sum_{i=1}^{N_{\text{Fit}}} \frac{D_{i,s}}{D_{i,o}} \times 100\%, \quad (4)$$

where $D_{i,s} \leq D_{i,o}$, and they are the durations of synthesized waveforms and original waveforms, respectively.

- **D. Ex. (%)**: the average percentage of the **durations** for the synthesized waveforms that **exceed** the original durations, formulated as,

$$\sum_{j=1}^{N_{\text{Exceed}}} \frac{D_{j,s}}{D_{j,o}} \times 100\%, \quad (5)$$

where $D_{j,s} \geq D_{j,o}$.

Table 6 shows an example of measuring duration matching.

Our system meets the requirement by trying multiple translation candidates for each source sentence. In the fast mode, our system uses the best three candidates that are generated by a machine translation model. In the quality mode, our system employs three machine translation models, that is, nine translation candidates. Table 2 shows that by increasing the number of translation models, the Fit and D. Fit percentages increase and D. Ex. decreases percentage accordingly.

No.	Source Sentence	Translation	Time (s)		
			Start.	Play.	Delay
1	大学教育入門第九章 アカデミックプレゼンテーション	Introduction to University Education Chapter 9: Academic Presentation	1.8	11.1	9.3
2	パートフォーの講義になります	Part Four.	6.0	15.3	9.3
3	この講義ではプレゼンテーションの話方についてまず説明します	In this lecture, we'll start with a presentation.	9.2	18.5	9.3
4	まず事前練習は必ずしましょう	Be sure to do the pre-practice first.	15.3	24.7	9.4
5	お部屋で一人ででもいいのでまずしゃべってみることが大事です	You can do it alone in the room, so it's important to talk to them first.	18.3	29.8	11.5
Average					9.8

Table 5: Example of measuring latency. **Start** time and **Playback** time are measured at the beginning of sentences and translations, respectively.

No.	Source Sentence	Translation	Duration(s)		Dur. Match. (%)		
			Sour.	Trans.	Fit	D.Fit	D.Ex.
1	一般契約ができたのも毎回毎回七社とプレゼン合うんですよ	I was able to make a general contract, and each time I made a presentation with seven companies, right?	4.97	4.18	Yes	84.1	
2	スピードデートみたいな形で三十分から一時間ずつ会っていきんですよ	We meet for thirty minutes to an hour each time in the form of a speed date.	3.45	3.01	Yes	87.2	
3	そのときに僕は世界的な著者になる準備をしてきたし	That's when I was preparing to become a world-class author.	3.70	3.02	Yes	98.3	
4	日本でも実績もあるしほほいけるんじゃないかなと思うと	I also have a track record in Japan, so I think I'll be almost able to do it.	3.05	3.34	No		109.5
5	もちろん確信は100%あるわけじゃないけど僕はその仲間も助けてくれることもあるし	Of course, I'm not 100 percent sure, but sometimes my friends can also help me.	5.43	3.46	Yes	63.7	
Average					80.0	79.1	109.5

Table 6: Example of measuring duration matching.

	Buffer size threshold (seconds)										
	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
a. Latency (seconds)											
x1.00	14.66	14.66	14.66	14.66	14.66	14.66	14.66	14.66	14.66	14.66	14.66
x1.01	13.22	13.22	13.23	13.26	13.32	13.41	13.54	13.72	13.92	14.13	14.30
x1.02	12.59	12.59	12.61	12.66	12.74	12.87	13.05	13.28	13.54	13.83	14.08
x1.03	12.19	12.20	12.23	12.28	12.39	12.54	12.75	13.00	13.31	13.64	13.94
x1.04	11.90	11.91	11.95	12.02	12.13	12.31	12.53	12.82	13.14	13.50	13.83
x1.05	11.69	11.70	11.73	11.81	11.94	12.13	12.37	12.67	13.02	13.39	13.74
x1.06	11.50	11.52	11.56	11.64	11.78	11.98	12.24	12.55	12.92	13.30	13.67
x1.07	11.35	11.37	11.41	11.50	11.65	11.86	12.13	12.46	12.83	13.23	13.61
b. # stalls (per minute)											
x1.00	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.42
x1.01	0.66	0.68	0.66	0.64	0.62	0.58	0.53	0.51	0.47	0.45	0.43
x1.02	0.98	1.00	0.95	0.91	0.84	0.74	0.66	0.58	0.52	0.48	0.45
x1.03	1.28	1.28	1.22	1.14	1.02	0.90	0.77	0.65	0.56	0.50	0.46
x1.04	1.54	1.53	1.45	1.34	1.19	1.03	0.87	0.72	0.61	0.52	0.48
x1.05	1.83	1.80	1.66	1.52	1.34	1.14	0.94	0.78	0.64	0.53	0.49
x1.06	2.11	2.08	1.91	1.70	1.49	1.24	1.02	0.82	0.67	0.56	0.50
x1.07	2.41	2.32	2.12	1.87	1.60	1.34	1.08	0.87	0.69	0.57	0.50
c. Total duration of stalls (seconds per minute)											
x1.00	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27
x1.01	1.69	1.67	1.66	1.62	1.58	1.52	1.46	1.40	1.35	1.32	1.30
x1.02	2.20	2.17	2.11	2.03	1.91	1.78	1.65	1.53	1.43	1.36	1.32
x1.03	2.72	2.66	2.56	2.40	2.22	2.01	1.81	1.64	1.50	1.40	1.34
x1.04	3.24	3.14	2.98	2.76	2.50	2.22	1.96	1.73	1.56	1.43	1.36
x1.05	3.76	3.61	3.39	3.10	2.76	2.42	2.10	1.82	1.61	1.47	1.38
x1.06	4.26	4.07	3.78	3.42	3.01	2.60	2.22	1.91	1.66	1.49	1.39
x1.07	4.76	4.51	4.16	3.72	3.25	2.77	2.34	1.98	1.71	1.52	1.41

Table 7: Grid search for the optimal buffer size threshold (0.0 - 10.0 seconds) and playback acceleration (x1.00 - x1.07) for the fast mode. The criteria are: **a.** Latency is as small as possible. **b.** # stalls ≤ 3 times per minute. **c.** Total duration of stalls ≤ 3 seconds per minute.

Our system chooses the longest translated speech waveform within the original duration among the candidates. If all the waveforms exceed the original duration, our system will choose the shortest one and truncate its excess to avoid overlapping with the next sentence. Our system does not adjust speech rate as it makes the sound weird and degrades viewing experience.

We have tried controlling the output length of machine translation, similar to (Lakew et al., 2019), but for our Japanese-English language pair, the translation quality drops a lot. We think the reason is that these two languages are so different that the translation cannot be enforced to have a similar length with the source sentence.

5 Conclusion

This paper presents our Japanese-to-English simultaneous dubbing prototype. The system enables low-latency and smooth live video streaming in the target language. We believe this technology will find widespread use in global communications.

In the future, we plan to add optical character recognition to our system. Video streaming often displays some text, such as the slides that appear in a lecture. Text in video streaming is an important source of information for viewers. Therefore, we hope that by recognizing and translating the text in video streaming, our system can provide users with a complete viewing experience in the target language.

Acknowledgements

A part of this work was conducted under the commissioned research program “Research and Development of Advanced Multilingual Translation Technology” in the “R&D Project for Information and Communications Technology (JPMI00316)” of the Ministry of Internal Affairs and Communications (MIC), Japan.

Ethical Considerations

Our system differs from generating deepfake video contents. Viewers can distinguish the dubbed video streams from original video streams, so it is unlikely for others to use our system in harmful ways. The purpose of our system is to deliver information to viewers in their native language, not to generate realistic videos. We do not synchronize lip with speech or render speech with background noise because they would not help with that goal but introduce additional latency in the output. From these two aspects, viewers can tell the dubbed streams from original video streams. Additionally, we place visible annotations on the output stream indicating that it is dubbed by automatic machine translation.

References

- I Ahmad Abuarqoub. 2019. Language barriers to effective communication. *Utopía y Praxis Latinoamericana*, 24.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Proceedings of the 3rd International Conference on Learning Representations.*, pages 1–15.
- Christos George Bampis, Zhi Li, Anush Krishna Moorthy, Ioannis Katsavounidis, Anne Aaron, and Alan Conrad Bovik. 2017. Study of temporal effects on subjective video quality of experience. *IEEE Transactions on Image Processing*, 26(11):5217–5231.
- Ondřej Bojar, Dominik Macháček, Sangeet Sagar, Otakar Smrž, Jonáš Kratochvíl, Peter Polák, Ebrahim Ansari, Mohammad Mahmoudi, Rishu Kumar, Dario Franceschini, Chiara Canton, Ivan Simonini, Thai-Son Nguyen, Felix Schneider, Sebastian Stüker, Alex Waibel, Barry Haddow, Rico Sennrich, and Philip Williams. 2021. [ELITR multilingual live subtitling: Demo and strategy](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 271–277, Online. Association for Computational Linguistics.
- Marcello Federico, Robert Enyedi, Roberto Barra-Chicote, Ritwik Giri, Umut Isik, Arvinth Krishnaswamy, and Hassan Sawaf. 2020. [From speech-to-speech translation to automatic dubbing](#). In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 257–264, Online. Association for Computational Linguistics.
- Dario Franceschini, Chiara Canton, Ivan Simonini, Armin Schweinfurth, Adelheid Glott, Sebastian Stüker, Thai-Son Nguyen, Felix Schneider, Thanh-Le Ha, Alex Waibel, Barry Haddow, Philip Williams, Rico Sennrich, Ondřej Bojar, Sangeet Sagar, Dominik Macháček, and Otakar Smrž. 2020. [Removing European language barriers with innovative machine translation technology](#). In *Proceedings of the 1st International Workshop on Language Technology Platforms*, pages 44–49, Marseille, France. European Language Resources Association.
- Shoichi Furukawa, Takuya Kato, Pavel Savkin, and Shigeo Morishima. 2016. Video reshuffling: automatic video dubbing without prior knowledge. In *ACM SIGGRAPH 2016 Posters*, pages 1–2.
- M-N Garcia, Francesca De Simone, Samira Tavakoli, Nicolas Staelens, Sebastian Egger, Kjell Brunnström, and Alexander Raake. 2014. Quality of experience and http adaptive streaming: A review of subjective studies. In *2014 sixth international workshop on quality of multimedia experience (qomex)*, pages 141–146. IEEE.
- Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772. PMLR.
- Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.
- Tobias Hößfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. 2011. Quantification of youtube qoe via crowdsourcing. In *2011 IEEE International Symposium on Multimedia*, pages 494–499. IEEE.
- Javier Iranzo-Sánchez, Adria Giménez Pastor, Joan Albert Silvestre-Cerda, Pau Baquero-Arnal, Jorge Civera Saiz, and Alfons Juan. 2020. Direct segmentation models for streaming speech translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2599–2611.
- Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. 2018. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31.

- Jaehyeon Kim, Jungil Kong, and Juhee Son. 2021. [Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech](#). *CoRR*, abs/2106.06103.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senelart, and Alexander Rush. 2017. [OpenNMT: Open-source toolkit for neural machine translation](#). In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.
- Surafel Melaku Lakew, Mattia Di Gangi, and Marcello Federico. 2019. [Controlling the output length of neural machine translation](#). In *IWSLT 2019 International Workshop on Spoken Language Translation*.
- Anush Krishna Moorthy, Lark Kwon Choi, Alan Conrad Bovik, and Gustavo De Veciana. 2012. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6):652–671.
- Markus Müller, Thai Son Nguyen, Jan Niehues, Eunah Cho, Bastian Krüger, Thanh-Le Ha, Kevin Kilgour, Matthias Sperber, Mohammed Mediani, Sebastian Stüker, and Alex Waibel. 2016. [Lecture translator - speech translation framework for simultaneous lecture translation](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 82–86, San Diego, California. Association for Computational Linguistics.
- Ricardo R Pastrana-Vidal, Jean Charles Gicquel, Catherine Colomes, and Hocine Cherifi. 2004. Sporadic frame dropping impact on quality perception. In *Human Vision and Electronic Imaging IX*, volume 5292, pages 182–193. SPIE.
- Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. 2019. Wav2letter++: A fast open-source speech recognition system. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6460–6464. IEEE.
- Yining Qi and Mingyuan Dai. 2006. The effect of frame freezing and frame skipping on video quality. In *2006 international conference on intelligent information hiding and multimedia*, pages 423–426. IEEE.
- Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32.
- Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2014. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492.
- Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238.
- Felix Stahlberg. 2020. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.
- Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Tatiana Likhomanenko, Edouard Grave, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. 2019. End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*.
- 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*, pages 5998–6008.
- Xiaolin Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2016. A prototype automatic simultaneous interpretation system. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 30–34.
- Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Online sentence segmentation for simultaneous interpretation using multi-shifted recurrent neural network. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 1–11.
- Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. 2022. [A multimodal simultaneous interpretation prototype: Who said what](#). 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 132–143, Orlando, USA. Association for Machine Translation in the Americas.
- Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. 2017. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*.
- Bartholomäus Wissmath, David Weibel, and Rudolf Groner. 2009. Dubbing or subtitling? effects on spatial presence, transportation, flow, and enjoyment. *Journal of Media Psychology*, 21(3):114–125.
- Yi Yang, Brendan Shillingford, Yannis M. Assael, Miaosen Wang, Wendi Liu, Yutian Chen, Yu Zhang,

Eren Sezener, Luis C. Cobo, Misha Denil, Yusuf Aytar, and Nando de Freitas. 2020. [Large-scale multilingual audio visual dubbing](#). *CoRR*, abs/2011.03530.

Wei Zhou, Xiongkuo Min, Hong Li, and Qiuping Jiang. 2022. A brief survey on adaptive video streaming quality assessment. *Journal of Visual Communication and Image Representation*, page 103526.

Alp Öktem, Mireia Farrús, and Antonio Bonafonte. 2019. [Prosodic Phrase Alignment for Machine Dubbing](#). In *Proc. Interspeech 2019*, pages 4215–4219.



VisKoP: Visual Knowledge oriented Programming for Interactive Knowledge Base Question Answering

Zijun Yao^{1*} Yuanyong Chen^{1*} Xin Lv¹ Shulin Cao¹ Amy Xin¹ Jifan Yu¹
 Hailong Jin¹ Jianjun Xu² Peng Zhang^{1,3} Lei Hou^{1†} Juanzi Li¹

¹Department of Computer Science and Technology,
 Tsinghua University, Beijing 100084, China

² Beijing Caizhi Technology Co., Ltd. ³ Zhipu.AI
 yaozj20@mails.tsinghua.edu.cn, houlei@tsinghua.edu.cn

Abstract

We present Visual Knowledge oriented Programming platform (VisKoP), a knowledge base question answering (KBQA) system that integrates human into the loop to edit and debug the knowledge base (KB) queries. VisKoP not only provides a neural program induction module, which converts natural language questions into knowledge oriented program language (KoPL), but also maps KoPL programs into graphical elements. KoPL programs can be edited with simple graphical operators, such as “dragging” to add knowledge operators and “slot filling” to designate operator arguments. Moreover, VisKoP provides auto-completion for its knowledge base schema and users can easily debug the KoPL program by checking its intermediate results. To facilitate the practical KBQA on a million-entity-level KB, we design a highly efficient KoPL execution engine for the back-end. Experiment results show that VisKoP is highly efficient and user interaction can fix a large portion of wrong KoPL programs to acquire the correct answer. The VisKoP online demo¹, highly efficient KoPL engine², and screencast video³ are now publicly available.

1 Introduction

Knowledge Base Question Answering (KBQA) aims to find answers to factoid questions with an external Knowledge Base (KB). Researchers have fully explored the KBQA (Lan et al., 2021) task and the most common solution is to convert user-posed natural language questions into KB query programs via semantic parsing and then give a final result by executing queries on the KB, such as SPARQL (Mihindukulasooriya et al., 2020; Gu et al., 2021), λ-DCS (Wang et al., 2015; Shin et al.,

* Equal contribution.

† Corresponding author.

¹demoviskop.xlore.cn (Stable release of this paper) and viskop.xlore.cn (Beta release with new features).

²<https://pypi.org/project/kopl-engine>

³<https://youtu.be/zAbJtxFPTXo>

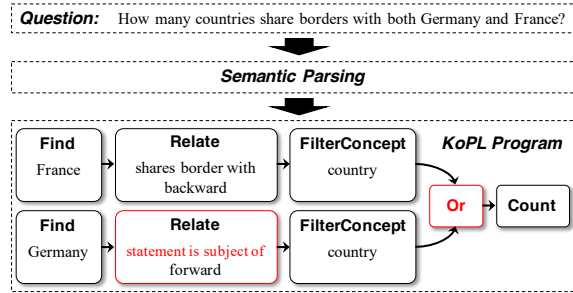


Figure 1: Semantic parsing results for natural language question “How many countries share borders with Germany and France?” given by state-of-the-art model trained on KQA Pro. Errors are marked in red color.

2021), and KoPL (Cao et al., 2022a,b). Recently, many KBQA systems (Höffner et al., 2013; Cui et al., 2016; Abdelaziz et al., 2021; Chen et al., 2021) that implement those advanced semantic parsing algorithms in an online environment, have been developed.

Although semantic parsing methods have gained considerable achievement, there is still no guarantee to precisely parse every user-posed question given the limitations of current machine learning techniques. Figure 1 demonstrates an example of semantic parsing results by the state-of-the-art KBQA model (Cao et al., 2022a). As the posed question does not follow the identical distribution of the training dataset adopted by the semantic parsing model (Shaw et al., 2021; Yin et al., 2021), it is falsely parsed with the *Or* operator, which should be an *And* operator, causing the **structure error** of the KB query. Meanwhile, it is extremely difficult for the semantic parsing model to correctly predict all the knowledge elements in a question (Cao et al., 2022b). As shown in the example, the “shares border with” relation is falsely predicted as a “statement is subject of” relation, causing an **argument error** in the KB query. However, existing KBQA systems do not provide easy access to manipulating the KB query programs and thus users cannot

intervene in the query execution.

Fortunately, several program-based KB query languages for complex reasoning consisting of modularized operators have come up, making KBQA easy to visualize (Ansari et al., 2019; Saha et al., 2019). With applicable visual representation of KB queries, intended users are capable of identifying errors in the programs generated by semantic parsing and correct them. Based on these observations, we raise a natural question: *How to design a visualized KBQA system that eases users to inspect and debug those KB query programs?*

Presented System. We demonstrate **Visual Knowledge oriented Programming (VisKoP)** platform, an interactive, visualized and program-based KBQA system. VisKoP provides an interactive knowledge oriented programming environment, allowing users to monitor and debug the KB queries with graphical operators. In comparison with existing KBQA systems, VisKoP is easier to use due to its following characteristics:

- **Knowledge oriented programming.** VisKoP is the first KBQA system to support Knowledge oriented Programming Language (KoPL) (Cao et al., 2022a). As a program-based KB query language, KoPL provides modularized program style for users to interact with knowledge elements, within its wide range of knowledge operators. Besides, KoPL can be converted into various different KB query languages via GraphQL IR (Nie et al., 2022).
- **Visualized interface.** VisKoP maps programming with KoPL into a series of graphical operations—“*dragging*” to add new knowledge operators, “*connecting*” the knowledge operators to add dependencies, and “*slot-filling*” to specify knowledge arguments.
- **Interactive programming and debugging.** We use semantic parsing algorithms to convert natural language questions into KB queries, whose execution gives not only the final answers, but also intermediate results of each knowledge operator, which facilitates debugging. Meanwhile, auto-completion for KB schema (*e.g.*, relation, concept, and attribute) provided by VisKoP assists users that are unfamiliar with the KB schema.
- **High efficiency.** We develop a high performing KoPL engine for VisKoP’s back-end. It executes KoPL on a million-entity level KB in less than

200 milliseconds, which can hardly be sensed next to the network latency.

We conduct user study and find that with the help of the visualized programming interface, users can find the correct answer in an average 110.6 seconds, which alleviates the problem caused by error-prone semantic parsing algorithms. Meanwhile, our efficiency study shows that the execution engine is significantly faster than the original KoPL engine and Virtuoso by $16\times$ and $5\times$, respectively.

Contributions. (1) We design a visualized knowledge oriented programming platform for KBQA, which integrates human into the loop to write and debug KB queries. (2) We implement a high performing KoPL execution engine that scales KoPL to an up-to-date million-entity-level KB.

The development and deployment of VisKoP validates the effectiveness of allowing questioners to monitor the error in the KB queries. The visual programming platform provides external human guidance on the neural program induction model, and potentially improves the robustness the system.

2 Preliminaries

2.1 Knowledge Base

As defined by KoPL (Cao et al., 2022a), KB consists of 4 kinds of basic knowledge elements:

Entities are unique objects that are identifiable in the real world, *e.g.*, *Germany*.

Concepts are sets of entities that have some characteristics in common, *e.g.*, *Country*.

Relations depict the relationship between entities or concepts. Entities are linked to their concepts via relation *instance of*, while concept hierarchy is organized via relation *subclass of*.

Attributes link entities to data value descriptions, *e.g.*, *day of birth*. Attributes can be further classified into 4 types: date, year, string, and numbers.

These knowledge elements are further organized into 3 kinds of structured representation in triplets:

Relational knowledge are triplets organized as (head entity, relation, tail entity).

Literal knowledge are triplets organized as (entity, attribute, value).

Qualifier knowledge are bound with relational or literal knowledge to specify under which condition they are true. The qualifiers are organized as (relational/attribute knowledge, attribute, value).

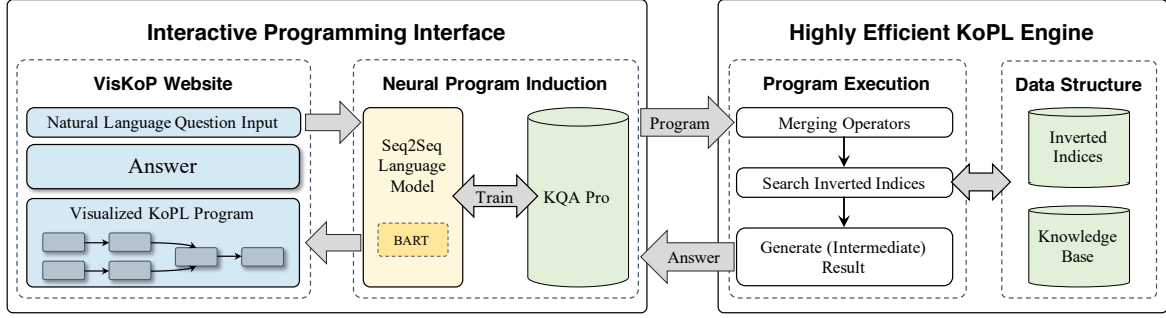


Figure 2: The overall system architecture of VisKoP.

2.2 Knowledge Base Question Answering

KBQA provides a natural-language-based interface for users to access knowledge in the KB. It inputs a natural language question $q = \{q_1, \dots, q_n\}$, where q_i is the i^{th} word, and outputs the answer utterance a . The answer is either the knowledge elements (e.g., entity name) in the KB, or the result of a combination of logical or algebraic operations performed on the knowledge elements.

2.3 KoPL

KoPL stands for knowledge oriented programming language consisting of 26 different knowledge operators. Natural language questions can be presented as KoPL programs, which are constructed as connected knowledge operators. Each operator has two categories of input: operator argument(s), and dependency input(s). Operator arguments are instructions on how to perform the operator, which are usually determined by the semantics of the question; Dependency inputs are outputs of previous knowledge operators that are linked to the current operator. For example, in Figure 1, operator *Relate(shares border with, forward)* has two arguments—*shares border with* and *forward*, while the dependency input comes from the *Find* operator. KoPL programs can be executed on the background KB to obtain the answer. More details are included in Appendix A.

One essential characteristic of KoPL is that, as modularized knowledge operators, the intermediate result of each operator is preserved and can thus be inspected and debugged. Given the modularity and inspectability of KoPL, we design the VisKoP platform, as described below.

3 The VisKoP Platform

The implementation of our VisKoP platform focuses on 4 designing principles:

I. Graphical Element Visualization: User-posed questions should be parsed into the KoPL program, and shown as graphical elements.

II. Interactive Programming: The system needs to enable users to edit and correct the KoPL program with knowledge schema auto-completion and intermediate results demonstration.

III. Highly Efficient Execution: The system should support large scale KBs for practical usage with low execution latency.

IV. Transparent Execution: The execution footprint of each operator should be preserved for inspection within interactive programming.

In particular, the first two principles are undertaken by the interactive programming interface in the front-end and the last two principles are undertaken by the highly efficient KoPL program execution engine in the back-end. The overall architecture of VisKoP is shown in Figure 2.

The implemented VisKoP is deployed as an openly available website¹. The highly efficient KoPL execution engine is also provided as an open-source Python extension toolkit².

3.1 Interactive Programming Interface

Graphical Element Visualization. VisKoP allows users to ask natural language questions and parse them into KoPL programs instead of writing KoPL programs from scratch. The process is carried out by a neural program induction module, as shown in Figure 2, whose backbone is a sequence-to-sequence pre-trained language model. Here we choose BART (Lewis et al., 2020) as the backbone and fine-tune it on the KQA Pro dataset (Cao et al., 2022a). It accepts natural language questions as input, and output the KoPL program in the depth first search order. The KoPL programs are converted to meet the format of sequence generation.

VisKoP visualizes KoPL program as a tree structure in the editing panel, where the nodes in the

tree are knowledge operators with arguments. Argument inputs are modeled as filling slots in the knowledge operators and dependency inputs are modeled as directed edges between different knowledge operators. We define 3 kinds of graphical actions that users may take within the KoPL program: *dragging* to add new operators, *linking* to indicate knowledge elements flow, and *slot-filling* to designate arguments of the knowledge operators.

Interactive Programming. For users that are less skilled at KoPL programming or less familiar with the schema of the underlying KB, VisKoP implements a series of auxiliary functions. Firstly, the KB schema is mainly associated with arguments of the knowledge operators. VisKoP helps to auto-complete knowledge elements via string matching when users try to fill in the argument slots. Next, to ensure the grammatical correctness of the KoPL program whose users submit to run, we implement linking legitimacy checking. VisKoP warns users when the submitted program is not a tree or the dependency is illegal (e.g., The output of the *Count* operator cannot be input to the *QFilterStr* operator). Finally, intermediate execution results of each knowledge operator are returned from the back-end and presented on the visualized interface where users may debug their KoPL program.

3.2 Highly Efficient KoPL Engine

The highly efficient KoPL engine is responsible for most parts of the back-end by reading the KoPL program as input and outputting the answer.

Highly Efficient Execution. KoPL program execution should be highly efficient for supporting large-scale KBs. Towards this goal, we adopt three implementation strategies: inverted indices construction, knowledge operators merging, and data structure optimization.

The first step is to construct inverted indices, which maps different types of attribute values and relations to their involved entities. These inverted indices prevent knowledge operators from enumerating over all the entities in the KB to recall corresponding knowledge elements. Subsequently, the great deal of time consumed by the engine to filter out entities satisfying certain constraint from the overall KB comes to our attention. This is represented by consecutive *FindAll* operator and filtering operators (e.g., *FilterStr*). We propose to merge the two consecutive operators and construct corresponding inverted indices. Finally, for all key-value



pair data structures, we use the running time of the questions in the KQA Pro dataset on the million-entity level KB as the metric, to greedily search out the optimal storage structure. The searching space contains hash map, red-black tree, trie tree, and ternary search tree.


Transparent Execution. Showing the intermediate results in the front-end requires the execution engine to preserve the outputs of each operator in the KoPL program and use them to monitor the behavior of the knowledge query. Meanwhile, users can debug the input KoPL program by inspecting the intermediate results to locate the bug.

4 Usage Example

4.1 Interactive Programming Interface

The online website of VisKoP is illustrated in Figure 3. We give an example of how to interact with the system to obtain the correct answer by questioning “How many countries share borders with both Germany and France?”, which cannot be correctly parsed by the semantic parsing algorithm.

Neural program induction. VisKoP accepts KB queries in natural language. The users input the question in the input box on the top of the website. Clicking on the  button parses the natural language question into its corresponding KoPL program, to be displayed on the editing panel at the bottom of the website. The predicted answer is shown by clicking the  button in the top of the editing panel. Here, VisKoP provides the common functionality as a KBQA system.

KoPL program debugging. As shown by Figure 3, users can easily identify two errors. One issue comes from the structural aspect. The answer should be counted on the intersection of two sets of country, each sharing border with Germany and France, respectively. To replace the operator *Or* with the operator *And*, users may first click on the *Or* operator for selection, and then press the backspace key to delete it. The *And* operator is added by selecting *Add* in the drop-down box and clicking on the  button. By linking the new operator to its dependency operators and the output operator, users can easily fix the structural error.

The other issue is the falsely recognized argument for the *Relate* operator. The desired countries should share borders with *Germany*, rather than the *statement is subject of* relation. The relation name specified by the KB schema is auto-completed in the pop-up drop down box, as shown in Figure 4.

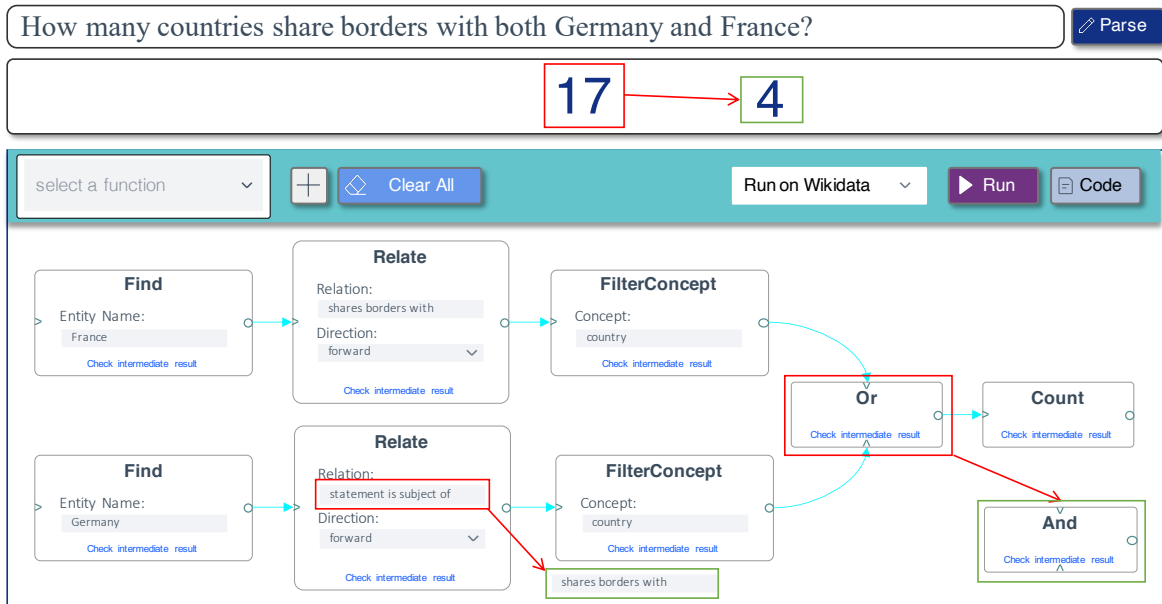


Figure 3: Screenshot of the interactive programming interface of VisKoP. When user tries to parse “How many countries share borders with both Germany and France?”, the semantic parsing algorithm falsely predict the *Or* operator, and one of the argument inputs of the *Relate* operator. This further results in the wrong answer “17”. We marked this errors in the red box, and put the correct graphical elements in the nearby green box.



Figure 4: Left: Screenshot of the auto-completion in slot-filling. Right: Screenshot of the intermediate result of the *And* operator, which shows the satisfied countries.

The intermediate result of each knowledge operator is a powerful tool to diagnose the KoPL program. It also serves as an interpretation to the question’s answer. By expanding the intermediate result of the *And* operator, as shown in the right part of Figure 4, we are able to know which countries are taken into account.

4.2 KoPL Engine

The high performing KoPL engine incorporated in the back-end is developed as an independent extension for Python. It provides one line installation code from the command line by running “pip install kopl-engine”. Users can execute the KoPL program using the scripts provided at the end of this section.

Users are first required to provide the KB in JSON file per the request by KoPL⁴. The execution

⁴https://kopl.xlore.cn/en/doc/4_helloworld

engine is initialized by converting the KB into data structure in the memory and constructing all the indices. Before executing the KoPL program, the engine parses the program represented in Python data structure (See Appendix B for the data structure introduction.) into the data structure used inside the engine. After that, users can call the forward method of the engine to get execution results.

```

1 from kopl_engine import engine
2 # Knowledge base preparation
3 kb = engine.init("kb.json")
4 # Data structure conversion
5 p = engine.parse_program(program)
6 # Program execution with
7 # intermediate result tracing
8 result = engine.forward(
9     kb, p, trace=True)

```

5 Evaluation

We evaluate the execution efficiency of the back-end KoPL engine. We also perform user study and case study to examine the limitations of VisKoP.

5.1 Efficiency

KB preparation. VisKoP is deployed on a million-entity-level KB extracted from Wikidata. In particular, we use the original Wikidata dump⁵ and only

⁵<https://dumps.wikimedia.org/wikidatawiki/entities/latest-all.json.bz2>

keep the entities that have a Wikipedia page. The statistics is shown in Table 1.

# Entity	# Concept	# Relation	# Attribute
6,284,269	68,261	1,080	1,352

Table 1: Statistics of the knowledge base.

Experimental setup. We use the training data of KQA Pro (Cao et al., 2022a) as the test-bed, which contains 94,376 queries in both KoPL and SPARQL program. We compare VisKoP against the original KoPL engine released by Cao et al. (2022a). We also compare it with Virtuoso for the SPARQL queries. All experiments are conducted on a single Intel Xeon 5218R CPU with 1.0TB RAM. We use wall time as the comparison metric.

Engine	VisKoP	KoPL	Virtuoso
Wall Time	111.5 ms	1775.8 ms	535.1 ms

Table 2: Running time averaged over all the queries.

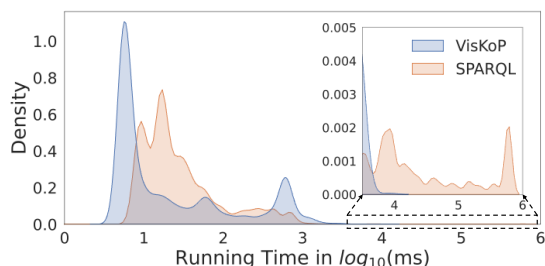


Figure 5: Running time distribution.

The averaged running time is reported in Table 2. VisKoP is almost $16\times$ faster than the original KoPL engine and $5\times$ faster than Virtuoso executing equivalent SPARQL queries. We also show the running time distribution of VisKoP and Virtuoso in Figure 5. VisKoP is faster than Virtuoso because: (1) The distribution peak of VisKoP comes smaller than Virtuoso; (2) The maximum running time of VisKoP is much smaller than Virtuoso.

5.2 User Study and Case Study

We manually annotate 20 natural language questions which cannot be correctly answered without user correction and ask 6 different users to use VisKoP to find the answer. After users interact with VisKoP, the accuracy rate reaches 65.8%, with an average of 110.7 seconds per question and a median of 68.0 seconds. These results indicate that

integrating human into the loop significantly broadens the boundaries of the KBQA system’s capabilities. Meanwhile, apart from knowledge elements not included in the KB, there are still questions that are extremely difficult to answer due to their obscure knowledge elements. For example, to answer “How many video game is SONY published in 2020?”, one need to find the *Sony Interactive Entertainment* entity rather than the *Sony*, which also occurs in the KB and our testers can hardly find the *Sony Interactive Entertainment* entity.

6 Related Works

In general, KBQA methods can be grouped into two categories: 1) semantic parsing (Berant et al., 2013; Yih et al., 2015; Cheng et al., 2017; Liang et al., 2017; Ansari et al., 2019; Cao et al., 2022b), which translates natural language questions into logical forms, whose execution on the KB achieves the answer; 2) information retrieval (Bordes et al., 2014; Xu et al., 2016; Miller et al., 2016; Shi et al., 2021; Zhang et al., 2022), which ranks the entities from the retrieved question-specific sub-KB to get the answer. Our VisKoP falls into the semantic parsing category. Specifically, VisKoP translates a question into the multi-step program, pertaining to the neural program induction (NPI) paradigm (Lake et al., 2015; Neelakantan et al., 2017; Liang et al., 2017; Wong et al., 2021; Cao et al., 2022a).

The main challenge of NPI is that question-program parallel data are expensive to obtain and the program’s huge search space makes the learning challenging. Existing works tackle this issue only by learning from question-answer pairs with various reinforcement learning techniques (Liang et al., 2017; Saha et al., 2019) or synthesizing question-program data to alleviate the data scarcity problem (Cao et al., 2022a; Gu et al., 2021). In this paper, our VisKoP proposes a different solution to this task by integrating humans into the program induction loop, providing external human guidance to program induction model, and potentially improving the system robustness.

Compared with other KBQA systems, including ReTraCk (Chen et al., 2021), SEMPRE (Berant et al., 2013), TRANX (Yin and Neubig, 2018), DTQA (Abdelaziz et al., 2021), our VisKoP is the first to enable users to interact with the system via a visual platform and intermediate results checking.

7 Conclusion and Future Work

We demonstrate VisKoP, a KBQA platform that allows users to monitor, edit, and debug KB queries. VisKoP is also accompanied with a highly efficient engine that scales KoPL execution to a million-entity-level KB. In the future, it is intriguing to allow users to customize the KB. It is also important to provide guidance for users to recognize the true knowledge elements in the large scale KB.

Limitations

As a KBQA system, VisKoP is still highly dependent on the correctness and broad knowledge coverage of the background KB. It is extremely difficult to find the correct answer when the relevant knowledge elements are unincluded or incorrect in the KB. Also, if there are confusing knowledge elements, as we mention in Section 5.2 that users can hardly identify the *Sony Interactive Entertainment* entity, it is difficult for users to correct the KoPL program.

Ethics Statement

Intended Use. VisKoP is designed for users to edit their knowledge base queries with graphical elements.

Potential Misuse. As we count, there are 339,531 human female entities and 1,458,903 male entities in total. It can lead to gender biased answers on the grounds that a number of females do not exist in the KB. This problem stems from the imbalanced data (Wikidata), and can be solved when Wikidata includes more female entities. Therefore, it's important to allow users to debug the knowledge base in future work.

Data. The VisKoP is built on a high-quality subset of Wikidata, which attributes to the intelligence of the crowd.

User Study. The participants in the user study part are volunteers recruited from graduate students majoring in engineering. Before the user study experiments, all participants are provided with detailed guidance in both written and oral form. The only recorded user-related information is usernames, which are anonymized and used as identifiers to mark different participants.

Acknowledgments

This work is supported by National Key R&D Program of China (2020AAA0105203), and a grant from the Institute for Guo Qiang, Tsinghua University (2019GQB0003)

References

- Ibrahim Abdelaziz, Srinivas Ravishankar, Pavan Kapapathi, Salim Roukos, and Alexander G. Gray. 2021. [A semantic parsing and reasoning-based approach to knowledge base question answering](#). In *AAAI*.
- Ghulam Ahmed Ansari, Amrita Saha, Vishwajeet Kumar, Mohan Bhambhani, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. [Neural program induction for KBQA without gold programs or query annotations](#). In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4890–4896. ijcai.org.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. [Semantic parsing on Freebase from question-answer pairs](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.
- Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. [Question answering with subgraph embeddings](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.
- Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. [KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base](#). In *ACL*.
- Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022b. [Program transfer for answering complex questions over knowledge bases](#). In *ACL*.
- Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. [ReTraCk: A flexible and efficient framework for knowledge base 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: System Demonstrations*, pages 325–336, Online. Association for Computational Linguistics.
- Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. [Learning structured natural language representations for semantic parsing](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

- Long Papers*), pages 44–55, Vancouver, Canada. Association for Computational Linguistics.
- Wanyun Cui, Yanghua Xiao, and Wei Wang. 2016. [KBQA: an online template based question answering system over freebase](#). In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4240–4241. IJCAI/AAAI Press.
- Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. [Beyond iid: three levels of generalization for question answering on knowledge bases](#). In *WWW'21*.
- K Höffner, C Unger, L Bühmann, J Lehmann, ACN Ngomo, D Gerber, and P Cimiano. 2013. [Tbsl question answering system demo](#). In *Proceedings of the 4th Conference on Knowledge Engineering Semantic Web*.
- B. Lake, R. Salakhutdinov, and J. Tenenbaum. 2015. [Human-level concept learning through probabilistic program induction](#). *Science*, 350:1332 – 1338.
- Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji rong Wen. 2021. [A survey on complex knowledge base question answering: Methods, challenges and solutions](#). In *IJCAI*.
- 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.
- Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. [Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.
- Nandana Mihindukulasooriya, Gaetano Rossiello, Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Mo Yu, Alfio Gliozzo, Salim Roukos, and Alexander Gray. 2020. [Leveraging semantic parsing for relation linking over knowledge bases](#). In *ISWC*.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. [Key-value memory networks for directly reading documents](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.
- Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. [Learning a natural language interface with neural programmer](#). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- Lunyu Nie, Shulin Cao, Jiaxin Shi, Jiuding Sun, Qi Tian, Lei Hou, Juanzi Li, and Jidong Zhai. 2022. [GraphQ IR: Unifying the semantic parsing of graph query languages with one intermediate representation](#). In *EMNLP*.
- Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. [Complex program induction for querying knowledge bases in the absence of gold programs](#). *Transactions of the Association for Computational Linguistics*, 7:185–200.
- Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. [Compositional generalization and natural language variation: Can a semantic parsing approach handle both?](#) 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 922–938, Online. Association for Computational Linguistics.
- Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. [TransferNet: An effective and transparent framework for multi-hop question answering over relation graph](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. [Constrained language models yield few-shot semantic parsers](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Yushi Wang, Jonathan Berant, and Percy Liang. 2015. [Building a semantic parser overnight](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.
- Catherine Wong, Kevin Ellis, Joshua B. Tenenbaum, and Jacob Andreas. 2021. [Leveraging language to learn program abstractions and search heuristics](#). In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11193–11204. PMLR.
- Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. [Question answering on Freebase via relation extraction and textual evidence](#).

- In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany. Association for Computational Linguistics.
- Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. [Semantic parsing via staged query graph generation: Question answering with knowledge base](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.
- Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. [Compositional generalization for neural semantic parsing via span-level supervised attention](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.
- Pengcheng Yin and Graham Neubig. 2018. [TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.
- Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong yan Chen. 2022. [Sub-graph retrieval enhanced model for multi-hop knowledge base question answering](#). In *ACL*.

A KoPL Definition

The functions used in this paper are the same as those mentioned in Cao et al. (2022a), so we will not devote a great deal of space for details. The specific meaning of each function can be found in (Cao et al., 2022a) or on our website ⁶. Here we only briefly introduce the philosophy of these operators:

Query Operators find and return the knowledge elements in the KB by matching their names. *e.g.*, *Find* returns the corresponding entities according to the input entity name.

Filter Operators take a set of knowledge elements as input, and keep the knowledge elements that satisfy the given conditions as output. *e.g.*, *FilterConcept* takes a set of entities as input and output entities that belong to a given concept.

Verification Operators are used to determine whether the output of the previous function has some relationship to the given value. This type of operators is often used to answer judgement questions. *e.g.*, *VerifyNum* can judge whether the function output is greater than (less than, equal to) a given value.

Selection Operators select some knowledge elements from the output of previous function under the given condition. *e.g.*, *SelectAmong* can select the entity with the largest or smallest value of an attribute from a given set.

Set Operators do inter-set operations on the output of two functions. *e.g.*, *And* can take the union of two sets.

B KoPL Program Format

In python, each knowledge operator is represented as a Dict in Python with three keys: `function` corresponds to the name of the knowledge operator. `inputs` corresponds to the argument inputs of the knowledge operator. And `dependencies` corresponds to the dependency inputs of the knowledge operator. For example, KoPL program in Figure 3 can be represented as:

```
1 program = {[
2   {
3     "function": "Find",
4     "inputs": ["France"],
5     "dependencies":[-1,-1]
6   },
7   {
8     "function": "Relate",
9     "inputs": ["shares border with", "
10    backward"],
11    "dependencies":[0]
12  },
13  {
14    "function": "FilterConcept",
15    "inputs": ["country"],
16    "dependencies":
17    [1]
18  },
19  {
20    "function": "Find",
21    "inputs": ["Germany"],
22    "dependencies":[]
23  },
24  {
25    "function": "Relate",
26    "inputs": ["statement is subject
27    of","forward"],
28    "dependencies": [3]
29  },
30  {
31    "function": "FilterConcept",
32    "inputs": ["country"],
33    "dependencies": [4]
34  },
35  {
36    "function": "Or",
37    "inputs": [],
38    "dependencies": [2,5]
39  },
40  {
41    "function": "Count",
42    "inputs": [],
43    "dependencies":[6]
44  }
45 ]}
```

C Questions for User Study

We list the 20 questions used in the user study and the corresponding answers in Table 3.

⁶https://kopl.xlore.cn/en/doc/2_function.html

Question	Correct Answer
How many Olympic Games has LeBron James competed in?	3
What is the name of the company that makes the game "The Legend of Zelda"?	Nintendo
How many teams have both LeBron and Kobe played for?	1
Is China more than 9.7 million square kilometres in size?	No
Which is the largest province in China by area?	Qinghai
How many countries are there in the European Union?	27
How many times has Federer won a tennis competition?	29
Which country is Google headquartered in?	United States of America
How many international airports are there in Germany?	15
Who is lighter, the iPhone X or the Samsung S10?	Samsung Galaxy S10
Which is the highest of all the mountains in South America and Africa?	Aconcagua
How many video game is SONY published in 2020?	566
Who is the next president after Barack Obama?	Donald Trump
At which college did Geoffrey Hinton get his degree?	University of Edinburgh
How many people have won the Nobel Prize in Physics?	186
What award did Lawrence win for The Hunger Games?	MTV Movie Award for Best Female Performance
How many states does the United States contain?	50
Which is the highest mountain in Asia?	Mount Everest
What year was the team owned by Jordan founded?	1988
In which country is Nikon headquartered?	Japan

Table 3: 20 questions used for user study.

PEEP-Talk: A Situational Dialogue-based Chatbot for English Education

Seungjun Lee¹, Yoonna Jang¹, Chanjun Park^{1,2}, Jungseob Lee¹
Jaehyung Seo¹, Hyeonseok Moon¹, Sugyeong Eo¹, Seounghoon Lee³
Bernardo Nugroho Yahya⁴, Heuseok Lim^{1*}

¹Korea University, South Korea, ²Upstage, South Korea

³Institute for Infocomm Research, A*STAR, Singapore

⁴Hankuk University of Foreign Studies, South Korea

{dzzy6505, morelychee, bcj1210, omanma1928, seojae777, glee889, djtnrud, limhseok}@korea.ac.kr

chanjun.park@upstage.ai, lees10i2r.a-star.edu.sg, bernardo@hufs.ac.kr

Abstract

English is acknowledged worldwide as a mode of communication. However, due to the absence of realistic practicing scenarios, students learning English as a foreign language (EFL) typically have limited chances to converse and share feedback with others. In this paper, we propose PEEP-Talk, a real-world situational dialogue-based chatbot designed for English education. It also naturally switches to a new topic or situation in response to out-of-topic utterances, which are common among English beginners. Furthermore, PEEP-Talk provides feedback score on conversation and grammar error correction. We performed automatic and user evaluations to validate performance and education efficiency of our system. The results show that PEEP-Talk generates appropriate responses in various real-life situations while providing accurate feedback to learners. Moreover, we demonstrate a positive impact on English-speaking, grammar, and English learning anxiety, implying that PEEP-Talk can lower the barrier to learning natural conversation in effective ways.

1 Introduction

In the era of globalization, English is used as a worldwide international language (Kramersch, 2014). A number of countries have incorporated the acquisition of foreign language communication skills into their lifelong learning objectives (Luna Scott, 2015). Altalbe and Wilkinson (2013) have identified several areas, including education, tourism, and business, where direct communication with people is crucial, and conversation skill is considered as the most essential of the various language competencies.

However, students learning English as a foreign language (EFL) do not have sufficient opportunities to practice real-life English conversations (Jdetawy, 2011). To address this issue, recent technologies such as smart speakers and conversational models

have been applied in English education (Tai and Chen, 2020; Alsadoon, 2021; Li et al., 2017). In particular, chatbots have shown promising results in improving the communication skills and learning motivation of students (Fryer and Carpenter, 2006).

Despite their potential, existing chatbot-based educational platforms face several challenges in providing effective language learning experiences. These chatbots employ hand-crafted and pattern-matching rules, limiting their communication ability and responding appropriately to out-of-situation utterances (Tye et al., 2022; Kim et al., 2019). Furthermore, as smart speakers and conversational AI models are not fully considered for educational purposes (Terzopoulos and Satratzemi, 2020; Ji et al., 2022), they cannot cover various topics in real-life activities or provide educational feedback.

To address these challenges, we propose PEEP-Talk, a situational dialogue-based chatbot for English education. It consists of a conversation module, context detector (CD), and grammar error correction (GEC) modules. The conversation module generates proper utterances considering the given situations with our proposed dataset, called SITUATION-CHAT. It covers a variety of real-world situations. To address the previous chatbots' inability to interact with dialogue out of topic or situation, the CD module changes the situation when the conversation digresses from the current situation. PEEP-Talk also provides feedback on learners' utterances with the situation similarity score, linguistic acceptability score, and grammar error correction.

We quantitatively verified the performance of each module of PEEP-Talk and conducted a user study to verify its effectiveness in English education in a real-world environment. To the best of our knowledge, there have been few attempts in NLP research to conduct a user study that verifies the performance and satisfaction of integrated modules. The comprehensive evaluation of PEEP-

Talk demonstrates the potential of our situational dialogue-based chatbot and feedback approach for effective English education. We deploy our methodology on a demo site. The code and dataset¹ are publicly available to facilitate further studies.

2 Chatbots for English Education

The field of language learning has been revolutionized by the emergence of chatbots. While previous computer-assisted language learning (CALL) systems have provided lessons and practice in writing, vocabulary, grammar, and other areas, they have limitations in that learners eventually need an actual human, such as a teacher or a colleague, to practice conversation (Chapelle, 2005). Chatbots have the potential to bridge this gap in CALL systems, as they offer more natural conversation and feedback similar to that of human instructors (Kim et al., 2019).

Commercial chatbots for English education have become increasingly popular, including Andy, Mondly Speak Now, Duolingo, and Babbel. Andy is a virtual tutor application that aids users in learning English, while Mondly features language lessons in various situations with a virtual teacher. Speak Now uses AI-driven virtual avatars as instructors for learning English. Duolingo offers gamified language lessons and has incorporated chatbots into its system, while Babbel features chatbots as part of its teaching methodology. The features of these existing chatbots for English education are also shown in Table 1.

However, one of the main problems with current English education chatbots is that they are unable to provide a genuinely free and natural conversation experience. Many chatbots rely on rule-based systems that are limited in generating diverse and spontaneous responses to user input (Fryer and Carpenter, 2006; Kim et al., 2019). As a result, learners often feel frustrated and disconnected from the language they are trying to learn (Fryer and Nakao, 2009). Additionally, these chatbots may struggle to understand idiomatic expressions, provide personalized feedback, and recognize situational contexts. Without engaging in meaningful and contextually relevant conversations, learners may struggle to develop the communication skills necessary for real-life interactions in English-speaking environments.

¹<https://github.com/metterian/peep-talk>

	features			URL
	AI-based	Situation	Feedback	
Andy	✗	✗	✗	andychatbot.com
Mondly	✗	✗	✗	mondly.com
Speak Now	✗	✓	✗	speaknow.ai
Duolingo	✓	✗	✓	duolingo.com
Babbel	✗	✗	✓	babbel.com
PEEP-Talk	✓	✓	✓	peeptalk.us

Table 1: Comparison of chatbots for learning English. AI-based (not rule-based), Situation (use of situational dialogues), and Feedback (provision of learner feedback)

3 PEEP-Talk

We introduce PEEP-Talk and discuss its motivation and unique features in §3.1. Then, we present the SITUATION-CHAT in §3.2, which contains diverse and situational expressions. The overall architecture of PEEP-Talk, including its three modules - conversation, context detector (CD), and GEC, is described in §3.3. Finally, §3.4 covers PEEP-Talk’s deployment.

3.1 Why PEEP-Talk?

While existing chatbot-based educational platforms have shown promising results in improving students’ communication skills and motivation to learn English (Jia and Ruan, 2008; Fryer and Carpenter, 2006; Haristiani, 2019), they still face several challenges in providing a practical language learning situation. One of the significant limitations of these chatbots is their inability to interact with utterances out of topic or situation. Additionally, smart speakers and conversational AI models are not fully optimized for educational purposes, as they cannot cover various topics in real-life activities or provide feedback on grammar errors.

To address these challenges, we propose PEEP-Talk, a situational dialogue-based chatbot for English education. PEEP-Talk is designed to generate contextually appropriate utterances using our proposed dataset, called SITUATION-CHAT. Furthermore, the context detector (CD) module enables PEEP-Talk to identify and adapt to changes in conversation topics, providing a more natural and engaging learning experience. The grammar error correction (GEC) module provides instant feedback to learners to improve their linguistic accuracy.

PEEP-Talk’s situational dialogue-based approach and dataset with diverse situations offer an effective language learning experience for EFL

learners. PEEP-Talk aims to address the limitations of existing chatbot-based educational platforms and provide an interactive and engaging English education environment.

3.2 The SITUATION-CHAT Dataset

This section presents the SITUATION-CHAT dataset, a situational dialogue-based dataset for English learning. An example of this dataset is shown in Figure 1. The dataset contains various situation dialogue such as asking for directions, talking with friends in school, and company interviews, among others. To construct our dataset, we follow the definition of a situational dialogue in Klinghoffer (2008). Situational dialogue is a learning approach where learners participate in role-playing through a routine activity, allowing for a more natural and engaging conversation. This approach has been shown to enhance the learners’ communication skills, as it allows them to practice real-life scenarios (Klinghoffer, 2008).

To construct our proposed dataset, We adopted AI hub²’s dialog corpus, a Korean-English translation corpus that contains conversations in a variety of contexts across multiple domains. This corpus is composed of division (domain), predefined situations, and dialogue history. To develop a situational dialogue-based conversational AI, the contextual information of predefined situations is insufficient. Therefore, we employ human annotators to write additional descriptions for the predefined situations. A details of this process is provided in Appendix A.

The dataset contains 16,298 dialogues with 65,192 utterances for the training set and 1,000 dialogues with 4,000 utterances for the validation and test sets, respectively. Each turn in the dataset has an average length of 14 words, and it includes 303 different situations covering various domains, including shopping, traffic, and travel. The statistical details of the dataset are written in Appendix A.3.

3.3 Overall Architecture

The architecture of PEEP-Talk consists of three modules: the conversation module, the context detector (CD) module, and the grammar error correction (GEC) module. The conversation module generates responses that are conditioned on the situation and the dialogue history. The CD module checks the user’s input to determine its appropri-

²The AI Hub, which can be accessed at <https://aihub.or.kr> is a public data platform operated by the government.

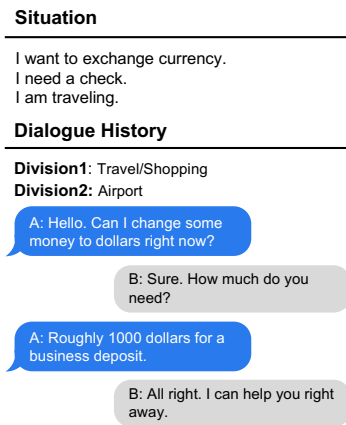


Figure 1: Example of the SITUATION-CHAT.

ateness in the topic or situation and provides feedback scores. The GEC module corrects grammatical errors in the conversation. Figure 2 provides an overview of the PEEP-Talk architecture.

3.3.1 Conversation Module

Compared to existing rule-based educational systems, the conversation module provides a more natural and diverse conversation experience to learners by employing DialoGPT (Zhang et al., 2019), a pre-trained language model trained on dialog corpus. It is further fine-tuned on our SITUATION-CHAT for learning situational expressions following the methodology presented in Wolf et al. (2019b). The input to the conversation module includes the situation, dialogue history, and candidate answers.

3.3.2 Context Detector Module

The primary objective of CD module is to provide feedback on learners’ responses when the conversation has deviated from the given situation. The main challenge that commercial chatbots face for English education is their inability to respond to inappropriate input from English beginners, including EFL learners (see Appendix H). Beginners often make linguistic errors or speak topics or situations that are out of topic in their conversations (Al Hosni, 2014; Sermsook et al., 2017), which the commercial chatbots are not equipped to handle appropriately. The CD module addresses this problem by checking the user’s utterance to determine its appropriateness in the topic or situation and whether it is linguistically natural.

The CD module gives situation similarity and linguistic acceptability scores on dialogue and switches dialogue situations. The situation similarity score measures the similarity of the user’s input

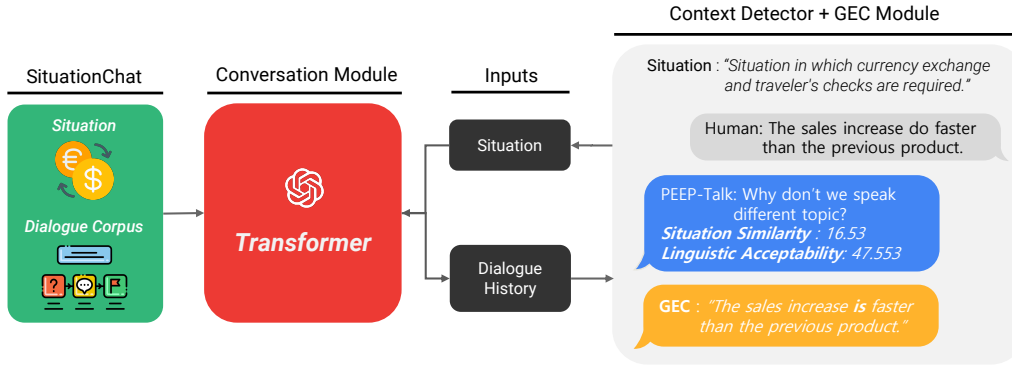


Figure 2: An overview of the modules in PEEP-Talk.

Models	Validation		Test	
	PPL	Hits@1	PPL	Hits@1
Profile Memory	42.79	0.078	45.02	0.069
Lost In Conversation	-	0.168	-	0.152
GPT _{SMALL}	12.41	0.849	12.74	0.839
GPT-2 _{SMALL}	12.50	0.839	12.56	0.848
DialoGPT _{SMALL}	12.35	0.850	12.55	0.856
DialoGPT _{MEDIUM}	14.77	0.884	13.89	0.864
DialoGPT _{LARGE}	11.15	0.889	12.04	0.877

Table 2: Experimental results for validation and test set of SITUATION-CHAT. PPL denotes perplexity and Hits@1 denotes the correct identification of the gold answer among 19 randomly sampled utterance candidates.

to the current situation (detailed in Appendix B), while the linguistic acceptability score checks the grammatical correctness of the user’s input. The CD module utilizes XLNet (Yang et al., 2019), fine-tuned on MRPC (Dolan and Brockett, 2005) and CoLA datasets (Wang et al., 2019), respectively.

The CD module suggests a new situation if the situation similarity score is under a certain threshold. Furthermore, as shown in Figure 2, learners can see the feedback score of the CD module during the conversation. Switching situation and real-time feedback helps to enhance the learner’s understanding of appropriate language use in different situations and promote their overall communication skills.

3.3.3 GEC Module

To provide accurate and efficient English grammar feedback to English learners, a deep learning-based GEC module has been integrated into PEEP-Talk in the form of a REST API. Specifically, we have adopted the approach described in Park et al. (2020), which uses a sequence-to-sequence (S2S) model. This approach is based on a noise sequence

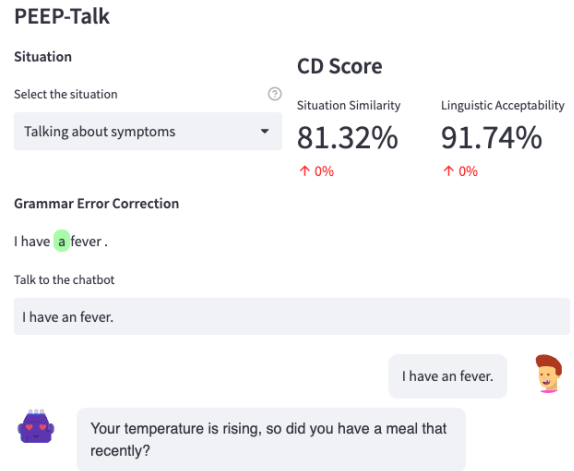


Figure 3: A screenshot of our demo website. Learners can choose various situations from the select box and receive responses. Grammatical errors are corrected and highlighted. The feedback scores are shown in the CD Score.

to denoise sequence (NS2DS) model, incorporating the copy mechanism (Gu et al., 2016). Notably, this method has been validated using commercial data and has been successfully applied in actual services. With this method, our GEC module ensures accurate and helpful real-time feedback on the learners’ grammar errors.

3.4 Deployment

The proposed PEEP-Talk is deployed through the web to render it easier for more people to access our system. We used HuggingFace (Wolf et al., 2019a) and TextAttack (Morris et al., 2020) library for developing modules. We use FastAPI to make all our modules as a REST API, then we build the web application³ with the Streamlit library. A screenshot of our demo website is shown in Figure 3.

³<http://peeptalk.us>

Model	Fluency	Situation Consistency	Engagingness
Human	4.54	4.51	4.63
Profile Memory	2.78	1.79	2.25
GPT-2 _{SMALL}	2.54	3.11	3.44
DialoGPT _{LARGE}	3.78	3.92	4.13

Table 3: Results of human evaluation of conversation module. Human indicates the gold utterance of the test set annotated by humans.

4 Experiments

In this section, the effectiveness of the conversation module is verified through several automatic evaluation metrics and human evaluation. Also, we analyze CD and GEC modules in Appendix D and Appendix D.3 respectively.

4.1 Verification of Conversation Module

4.1.1 Quantitative Analysis

Method To empirically verify the conversation module in situational dialogue, we compare existing baselines and our model with the ability to generate and select the proper utterances. We utilize Profile Memory (Dinan et al., 2019) and *Lost In Conversation* as our baselines. For our model, GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019) and DialoGPT (Zhang et al., 2019) are exploited and fine-tuned on our data. Experimental settings are described in Appendix C. Furthermore, perplexity, and Hits@1 score indicating the correct identification of gold answers among 19 randomly sampled utterance candidates are used.

Results The experimental results of the conversation module are presented in Table 2. The experiment is conducted on validation and test sets of SITUATION-CHAT. DialoGPT_{LARGE} exhibits the best performance with a perplexity of 12.04 and Hits@1 score of 0.877. This suggests that DialoGPT is more effective in generating and selecting appropriate responses in situational dialogue than the other models and baselines. Therefore, the results suggest that DialoGPT can be a promising model for situational dialogue. The experimental settings and details are provided in Appendix C.

4.1.2 Qualitative Analysis

Method We evaluate the effectiveness of PEEP-Talk in providing contextualized responses in various situations properly. The data of 30 situations are randomly selected from the test set of SITUATION-CHAT. For comparison with exist-

ing methodologies, we select DialoGPT_{LARGE} fine-tuned on our proposed dataset, DailyDialog (Li et al., 2017), and the English education chatbots Andy and Speak Now.

Results Table 4 presents the results of the qualitative analysis. In the “Asking for directions” situation, PEEP-Talk gives a specific and helpful response, while DailyDialog’s response lacks useful information. In “Requesting packaging”, PEEP-Talk informs the customer about the availability of containers and demonstrates knowledge about the establishment, while DailyDialog’s response is unhelpful. In the “Stopping credit card” situation, PEEP-Talk is straightforward in asking for necessary information, whereas Andy provides an unsolicited introduction. In “Pharmacy”, PEEP-Talk provides a practical solution by recommending a medicine, while DailyDialog and Speak Now provide generic responses. PEEP-Talk shows that it can provide informative and helpful responses tailored to the situation at hand.

4.1.3 Human Evaluation

To validate the quality of the conversation module fine-tuned with SITUATION-CHAT, we conducted a human evaluation. First, fluency, situational consistency, and engagingness are measured as shown in Appendix E. We set the response level on a Likert scale (Likert, 1932) from 1 to 5. The evaluators are asked to evaluate the quality of the 150 randomly chosen model-generated utterances chosen from the test set of SITUATION-CHAT. The evaluators comprise English instructors, a teaching assistant, and three people with bachelor’s degrees in English literature⁴. Table 3 reports the evaluation results. The result shows that DialoGPT_{LARGE} exhibits the highest score in all factors and is the most similar to the human level, indicating the potential of our model for providing a more natural and engaging conversation experience for English learners.

5 User Evaluation

To verify the performance and satisfaction of PEEP-Talk as an English education tool, we conducted a user evaluation, which is rarely performed in NLP research. While other research often concludes their evaluation with human evaluation, we go further by actually operating our system and verifying its effectiveness in real-world English education.

⁴They were paid \$0.4 per dialogue evaluation.

Situation & Previous utterance	PEEP-Talk	DailyDialog	Andy	Speak Now
Asking for directions "Can you tell me how to get to the nearest train station?"	"You can take the train to the City Hall Station if you go out exit number 2 on the 2nd."	"I can't, I'm not a train driver."	"Okay"	"Good Man."
Requesting packaging "Can I get the leftovers to go?"	"You can, but we don't have a container for soup dishes."	"I'm sorry, I don't have any leftovers."	"Sure, will you write me later?"	"I'm looking at you"
Stopping credit card "Hello? I lost my credit card and I am calling to report the loss."	"Yes, please tell me your social security number and the card number of the person who used it."	"I lost my credit card and I am calling to report the loss."	"I see"	"Let's keep going."
Pharmacy "I caught a bad cold. I cough a lot and have a high fever."	"Take this cold medicine pill three times a day."	"I hope you feel better soon!"	"Got it"	"Oh, You hurt my feelings"

Table 4: Comparison of PEEP-Talk with other conversational agents in different situations

	Group	Mean	SD	Adj. Mean	$F(1, 32)$	p	
English Speaking	Accuracy	Exp.	2.41	0.51	2.58	0.77	0.18
		Ctrl.	2.67	0.62	2.64		
	Communication	Exp.	2.76	0.66	2.79	7.04*	0.03
		Ctrl.	2.93	0.70	2.69		
	Fluency	Exp.	2.93	0.62	2.81	3.6*	0.03
		Ctrl.	2.59	0.62	2.67		
	Comprehension	Exp.	3.35	0.49	3.47	3.98*	0.04
		Ctrl.	3.33	0.49	3.30		
	Maturity of the language	Exp.	2.71	0.59	2.98	12.52*	0.00
		Ctrl.	3.07	0.70	2.94		
Grammar	Exp	72.22	8.95	64.51	2.78*	0.04	
	Ctrl.	65.28	12.06	63.80			
Learning anxiety	Exp	2.89	0.33	3.04	2.92*	0.04	
	Ctrl.	3.08	0.61	3.14			

Table 5: Analysis of covariance of English-speaking performance, grammar, and learning anxiety for the two groups. Adj. Mean = adjusted mean. Exp. = experimental group with 18 people; Ctrl. = control group with 17 people. * $p < 0.05$.

Method The users evaluation consisting of assessments in English speaking, grammar, English learning anxiety, and user satisfaction. In order to conduct the evaluation, we divided participants into experimental and control groups, with 18 and 17 people respectively. They perform pre-test and post-test before and after practicing with our system. The experimental group used PEEP-Talk for 30 minutes per day for a period of two weeks, while the control group did not use any language learning tools during the evaluation period. Participants in both groups were consisted of women in their twenties who are native Korean speakers and university graduates. They are composed of those who speak EFL. The details of the experimental design can be found in Appendix F.

Results Based on the analysis of covariance (ANCOVA) results in Table 5, the experimental group shows better performance in most dimensions of English speaking, except for accuracy, indicating that PEEP-Talk improves learners' overall speak-

ing skills. In terms of grammar, the experimental group had a higher adjusted mean (64.51) compared to the control group (63.80), and the F-test result ($F(1, 32) = 2.78, p < 0.05$) indicates a significant difference in the grammar skills between the two groups. The English learning anxiety score was lower for the experimental group (adjusted mean = 3.04) than the control group (adjusted mean = 3.14). We conclude that PEEP-Talk has a positive impact on English-speaking, Grammar, and English learning anxiety after only two weeks of use.

5.1 PEEP-Talk Satisfaction Survey

We conduct satisfaction surveys to collect learners' satisfaction in a real English teaching environment. (presented in Appendix G). The survey results indicate that most users are satisfied with PEEP-Talk, with more than 70% of respondents finding the response of the chatbot natural and contextual. Moreover, 70 to 80% of respondents indicated that situational dialogue is educationally beneficial for conversational English and that grammar correction is helpful. However, the CD module receives a relatively low satisfaction rate of 46%, possibly due to the module switching the situation even when the user wished to continue talking about a specific situation.

6 Conclusion

In this paper, we propose PEEP-talk, an English education chatbot that allows learners to practice real-life conversations and receive feedback. Our research, pioneering in the realm of situational dialogue-based chatbots, was substantiated through rigorous qualitative and quantitative experiments. Furthermore, through user evaluations and satisfaction surveys, we confirmed that our method enhances educational efficiency and reduces learning anxiety for EFL learners in real-world English

teaching environments. This research is pivotal as it introduces a novel approach to chatbot-based English education, enhancing learning efficiency and engagement. In the future, we will also integrate additional features, such as learner progress tracking and an administrative dashboard for personalized instruction.

Limitations

This study has certain limitations. Firstly, the effectiveness of the CD module in detecting inappropriate input needs improvement. Secondly, the current version of PEEP-Talk covers only a limited number of topics and situations, warranting an extension of the dialogue dataset. Thirdly, the observed improvement in English speaking ability may not be entirely attributable to PEEP-Talk, as external factors such as daily English speaking practice outside the experiment could also have contributed. Moreover, the participant pool was relatively small, suggesting a need for future studies with larger and more diverse groups. Lastly, PEEP-Talk's current limitation to text-based interactions highlights an area for development, as incorporating voice interactions could further enhance the user experience. Future research could also investigate the correlation between linguistic acceptability and speaking and grammar performance.

Ethical Statement

We uphold the ethical principles outlined by the ACL Code of Ethics in our research. The public dataset used in our study was obtained from a nationally operated and managed AI Hub⁵. Prior to training the DialogGPT model, we followed the preprocessing steps outlined in Zhang et al. (2019) to remove offensive language using syntax matching against large blacklists and excluded subreddits that are likely to contain offensive content. To further prevent generating offensive content, we will exclude tokens in a list of several stopwords using Hatebase⁶. Furthermore, we ensure the privacy of our users by not collecting any personal data and only providing responses based on the situations that the learners have selected. We prioritize the ethical considerations of our study to maintain the integrity of our research and prevent any potential harm.

⁵<https://aihub.or.kr>

⁶<http://hatebase.org>

Acknowledgments

We would like to express our deep appreciation for the generous support we have received for our research. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) under grant number 2020-0-00368, titled "A Neural-Symbolic Model for Knowledge Acquisition and Inference Techniques". Additionally, this research was bolstered by the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency under grant number R2020040045. Lastly, we gratefully acknowledge the support of the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-2018-0-01405), which was supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

- Samira Al Hosni. 2014. Speaking difficulties encountered by young efl learners. *International Journal on Studies in English Language and Literature (IJSELL)*, 2(6):22–30.
- Reem Alsadoon. 2021. Chatting with ai bot: Vocabulary learning assistant for saudi efl learners. *English Language Teaching*, 14(6):135–157.
- Ali A Altalbe and Brett Wilkinson. 2013. Designing games for learning english as a second language. In *2013 International Conference on Computer Applications Technology (ICCAT)*, pages 1–7. IEEE.
- Carol A Chapelle. 2005. Computer-assisted language learning. In *Handbook of research in second language teaching and learning*, pages 767–780. Routledge.
- Shu-Yun Chien, Gwo-Jen Hwang, and Morris Siu-Yung Jong. 2020. Effects of peer assessment within the context of spherical video-based virtual reality on efl students' english-speaking performance and learning perceptions. *Computers & Education*, 146:103751.
- 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*.
- Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

- William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.
- L Fryer and K Nakao. 2009. Assessing chatbots for efl learner use. In *JALT2008 conference proceedings*, pages 849–857.
- Luke Fryer and Rollo Carpenter. 2006. Bots as language learning tools. *Language Learning & Technology*, 10(3):8–14.
- Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.
- Nuria Haristiani. 2019. Artificial intelligence (ai) chatbot as language learning medium: An inquiry. In *Journal of Physics: Conference Series*, volume 1387, page 012020. IOP Publishing.
- Elaine K Horwitz, Michael B Horwitz, and Joann Cope. 1986. Foreign language classroom anxiety. *The Modern language journal*, 70(2):125–132.
- Loae Fakhri Ahmad Jdetawy. 2011. Problems encountered by arab efl learners. *Language in India*, 11(3).
- Hyangeun Ji, Insook Han, and Yujung Ko. 2022. A systematic review of conversational ai in language education: focusing on the collaboration with human teachers. *Journal of Research on Technology in Education*, pages 1–16.
- Jiyoun Jia and Meixian Ruan. 2008. Use chatbot csiec to facilitate the individual learning in english instruction: A case study. In *International conference on intelligent tutoring systems*, pages 706–708. Springer.
- Na-Young Kim, Yoonjung Cha, and Hea-Suk Kim. 2019. Future english learning: Chatbots and artificial intelligence. *Multimedia-Assisted Language Learning*, 22(3):32–53.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Curtis L Klinghoffer. 2008. Situational dialogues in a community college: English as a second language curriculum. *Online Submission*.
- Claire Kramersch. 2014. Teaching foreign languages in an era of globalization: Introduction. *The modern language journal*, 98(1):296–311.
- 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*.
- Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.
- Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.
- 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*.
- Cynthia Luna Scott. 2015. The futures of learning 2: What kind of learning for the 21st century?
- John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.
- Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593.
- 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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.
- Chanjun Park, Yeongwook Yang, Chanhee Lee, and Heuseok Lim. 2020. Comparison of the evaluation metrics for neural grammatical error correction with overcorrection. *IEEE Access*, 8:106264–106272.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Kanyakorn Sermsook, Jiraporn Liamnimitr, and Rattaneekorn Pochakorn. 2017. The impact of teacher corrective feedback on efl student writers’ grammatical improvement. *English Language Teaching*, 10(10):43–49.
- Tzu-Yu Tai and Howard Hao-Jan Chen. 2020. The impact of google assistant on adolescent efl learners’ willingness to communicate. *Interactive Learning Environments*, pages 1–18.
- George Terzopoulos and Maya Satratzemi. 2020. Voice assistants and smart speakers in everyday life and in education. *Informatics in Education*, 19(3):473–490.

- Gladys Tyen, Mark Brenchley, Andrew Caines, and Paula Buttery. 2022. Towards an open-domain chatbot for language practice. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 234–249.
- 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 the Proceedings of ICLR.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019a. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019b. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.
- Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

A SITUATION-CHAT Dataset

A.1 Dialogue Corpus Collection

The dialogue corpus comprises division 1, division 2, predefined situations, and dialogue history. Divisions are domain information in a hierarchical structure, and predefined situation refers to a short description of the situation (e.g., “hospital admission situation”, “car accident situation”). Dialogue history is composed 4-turns dialogue of speaker A and B in each situation. We only use English data from the Korean-English parallel dataset. Further information regarding this corpus is provided in Table 6.

With the aid of a professional English instructor, we select situations and dialogue histories from the dialogue corpus, such as “taking a taxi” or “asking to take a picture”. Consequently, 330 situations out of 2,779 situations from dialogue corpus are selected with a total of 73,192 dialogue histories.

Division1	Domain information about dialogue sentences e.g.) Lectures, news, discussions, etc.
Division2	Specialty information to subdivide Division 1 e.g) Office, school, meetings, etc.
Predefined situation	One-line description of the situation e.g.) Exchange opinions
Dialogue history	Korean-English Conversation on Situations e.g.) How is the market’s reaction to the released product.

Table 6: Information and examples of dialogue corpus.

A.2 Situation Annotation

As shown in Figure 4, predefined situation in the dialogue corpus includes only one sentence to explain the situation. Therefore, the situational information is augmented with various virtual descriptions. For example, to describe the situation of currency exchange as shown in Figure 5, the dialogue history and the information of the domains are used to generate the sentences that describe the given situation. With four human annotators⁷, at least 2 and 5 sentences were generated for one predefined situation.

Specifically, we provide domain information, predefined situation, and conversation history and human annotators follow these rules when annotating the situations: (i) The subject of the sentence should be preferably in the first person. (ii) A description of the situation that should be inferred from the given information. In addition, they generated sentences in under 15 words.

⁷They were paid \$0.2 per one situation; they had a Bachelor’s degrees in English linguistics

Dialogue Corpus

Division1: Travel/Shopping

Division2: Airport

Predefined Situation : “Situation in which currency exchange and traveler’s checks are required.”

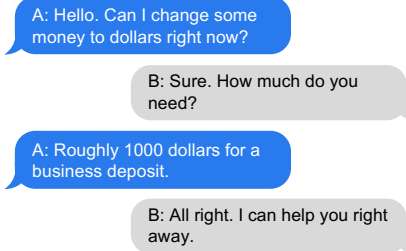


Figure 4: Example of dialogue corpus.

Annotated situation

I want to exchange currency.
I need a check.
I am traveling.

Figure 5: Example of annotated situation.

A.3 Dataset Statistics

The statistics of the entire dataset and domains are presented in Table 7 and Figure 6, respectively.

Category	Train	Valid	Test
Division 1	5	5	5
Division 2	38	32	30
# Situations	248	42	40
# Dialogues	16,298	1,000	1,000
# Utterances	65,192	4,000	4,000

Table 7: Statistics of SITUATION-CHAT. In case of the situations, they are separated into unseen ones for each other.

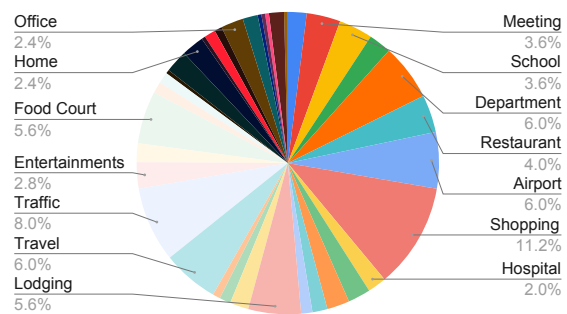


Figure 6: The domains of situations in SITUATION-CHAT.

B Situation Similarity Score

We utilize semantic textual similarity (STS) to measure the situation similarity score by comparing the user’s input and utterances of the predefined situation to determine it is out of situation (topic). Each predefined situation in SITUATION-CHAT contains an average of 480 utterances. The STS scores are computed with the user’s input and every utterance belong to the predefined situation dialogue data as shown in Figure 7. We use the maximum value of STS scores as the situation similarity score.

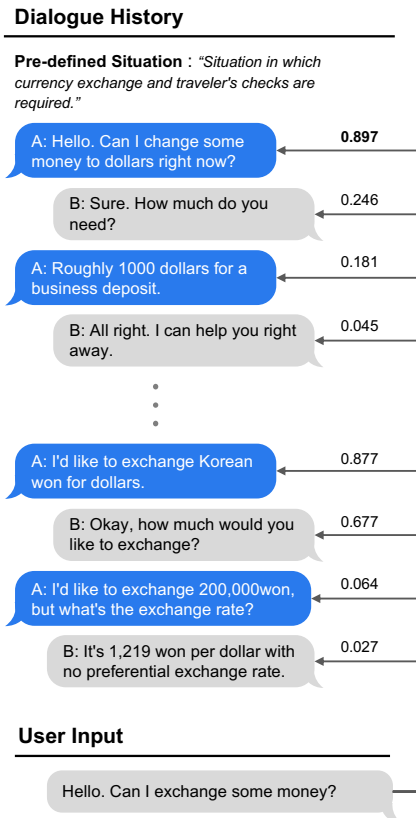


Figure 7: Example of scoring the situation similarity. We score the STS for pairs of the user’s input and utterances of the predefined situation in SITUATION-CHAT, then we consider the maximum score as the situation similarity. In the figure, the score represents the STS.

C Experimental Setting of Conversation Module

The experimental models used are based on ConVAI2’s methodologies, which are similar to the proposed approaches and publicly available. The baseline model of ConVAI2 is Profile Memory, while the state-of-the-art models are *TransferTransfo* and Lost In Conversation, which respectively achieved

the best results in the automatic and human evaluation.

Fine-tuning Details The detailed fine-tuning setting in this study is as follows. We trained our model on one GeForce RTX 8000 GPU for 18 hours with batch size of 8 and 2 epochs following the fine-tuning method of *TransferTransfo*(Wolf et al. (2019b)). We set the random seed as 42. Further, the Adam optimizer (Kingma and Ba, 2014) is employed and with learning rate set to converge from $6.25e - 5$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ to 0. Further, all layers are set with a dropout of 0.1 and a ReLU activation function. In terms of the number of model parameters, GPT has 117M, GPT-2_{SMALL} has 124M, DialoGPT_{SMALL} has 124M, DialoGPT_{MEDIUM} has 355M and DialoGPT_{LARGE} has 774M of trainable parameters.

Decoding details Beam search with the size of 4 and N-gram filtering methods are used during the generation process. The final beam is ranked according to the scala combination of the length-normalized utterance probability and next-utterance classification score. In addition, increasing the importance of the next-utterance classification score produces utterances more closely related to the conditioned situation sentences; however, the diversity of the dialogue is reduced.

D Verification of CD Module

The performance of the feedback on dialogue and the switching situation of CD module is verified. The experimental results of the two functions are shown in Table 9. The linguistic acceptability of feedback on dialogue is estimated using the validation and test set of CoLA. Further, the performance of switching situations using situation similarity of CD module is evaluated based on CEEU. Specifically, pre-trained transformers are adopted, and a test set of SITUATION-CHAT is evaluated with 1,000 situations in the experiments.

D.1 Experimental Setting

The CD module is trained on two datasets: Microsoft research paraphrase corpus (MRPC) and corpus of linguistic acceptability (CoLA). The training set of MRPC consists of 3,260 pairs, while the validation and test sets consist of 408 pairs respectively. For CoLA, the training set consists of 7,551 sentences, while the validation and test sets consist of 1,000 sentences each. These datasets are

used to evaluate the performance of the CD module in terms of situation similarity and linguistic acceptability.

We use TextAttack (Morris et al., 2020) for this experiment. We fine-tuned the MRPC dataset for situation similarity and the CoLA dataset for linguistic acceptability. We use pre-trained language models, which are BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019). We set the seed as 42. The experimental settings are presented in Table 8.

	Model	Epoch	Batch	LR	SeqLen	# Params
Situation	BERT	5	16	2e-05	256	110M
	ALBERT	5	32	2e-05	128	12M
	RoBERTa	5	16	3e-05	256	125M
	XLNet	5	32	5e-05	256	117M
Linguistic	BERT	5	32	2e-05	128	110M
	ALBERT	5	32	3e-05	128	12M
	RoBERTa	5	32	2e-05	128	125M
	XLNet	5	32	5e-05	128	117M

Table 8: We denote situation similarity as Situation, linguistic acceptability as Linguistic, the learning rate as LR, sequence length as SeqLen, and the number of trainable parameters as # Params.

Metric We propose CEEU metric to verify the situation switching function of the CD module. The conversation module generate the response based on dialogue history as shown in Figure 2. We input an utterance of out-of-situation in a random conversation turn in dialogue history, and measure whether the CD module detected it correctly.

Details of the corresponding procedures are described in Algorithm 1. SITUATION-CHAT denoted as D , contains a dialogue history $H_{n,t}$ for each situation n and a given turn t . Specifically, $H_{i,t}$ comprises gold answer $utt_{i,t}$, and utterance candidates $utt_{j,t}$ extracted from the different situation j . Once CD module correctly predicts the inclusion of the gold answer in a dialogue history, it is classified as a true positive (tp); otherwise, as a false negative (fn). Similarly, dialogue histories containing distractors are also classified into true negative (tn) or false positive (fp), based on the model judgment. Eventually, the accuracy is estimated based on the entire test set, where the number of utterances classified to the positive label and the negative label is set to be the same ratio.

D.2 Results

Table 9 presents the experimental results of the CD module, which are evaluated by the CEEU met-

Algorithm 1 CEEU

Require: $CD_{Module} = \{\text{MRPC}\}$

- 1: $S_n \leftarrow \{S_1, \dots, S_n \mid 1 \leq n \leq |D|\}$
- 2: /* S_n is the situation. */
- 3: /* $|D|$ is the number of dialogues. */
- 4: $t \in \{1, 2, 3\}$
- 5: $H_{n,t} \leftarrow \{utt_{n,1}, utt_{n,2}, \dots, utt_{n,t} \mid 1 \leq n \leq |D|\}$
- 6: /* utt_t is an utterance at turn t of n th dialogue. */
- 7: **procedure** CEEU($S_n, H_{n,t}$)
- 8: **for** $i \leftarrow 1, n$ **do**
- 9: $H_{i,t} \leftarrow H_{i,t-1} + \begin{cases} utt_{i,t} \\ \text{or} \\ utt_{j,t} \text{ where } i \neq j \end{cases}$
- 10: /* $utt_{j,t}$ is randomly chosen from other dialogues. */
- 11: ContextScore $\leftarrow CD_{Module}(S_i, H_{i,t})$
- 12: **if** $utt_{i,t}$ in $H_{i,t}$ **then**
- 13: **if** ContextScore ≥ 50 **then**
- 14: $tp \leftarrow tp + 1$
- 15: **else**
- 16: $fn \leftarrow fn + 1$
- 17: **else if** $utt_{j,t}$ in $H_{i,t}$ **then**
- 18: **if** ContextScore ≥ 50 **then**
- 19: $tn \leftarrow tn + 1$
- 20: **else**
- 21: $fp \leftarrow fp + 1$
- 21: **return** $Acc \leftarrow \frac{tp + tn}{tp + tn + fp + fn}$

ric for situation similarity and the CoLA dataset for linguistic acceptability. We fine-tune four pre-trained language models, including BERT, ALBERT, RoBERTa, and XLNet. The highest CEEU score is achieved by XLNet with a score of 0.628. The highest linguistic acceptability is achieved by BERT and XLNet with CoLA validation scores of 0.812 and 0.851 and CoLA test scores of 0.820 and 0.870, respectively.

Module	Model	CEEU	CoLA		GLUE	BLEU
			Valid	Test		
CD	BERT	0.476	0.812	0.820	-	-
	ALBERT	0.484	0.728	0.736	-	-
	RoBERTa	0.623	0.739	0.755	-	-
	XLNet	0.628	0.851	0.870	-	-
GEC	Park et al. (2020)	-	-	-	58.12	73.82

Table 9: Experimental results of CD module and GEC module. CEEU is used to evaluate situation similarity. We measure linguistic acceptability using valid and test set of CoLA. The performance of the GEC module is evaluated based on GLUE and BLEU.

D.3 Verification of GEC Module

The experimental result of GEC module is presented in Table 9. The performance of the GEC module is evaluated based on the GLUE (Napoles et al., 2015) and BLEU (Papineni et al., 2002) score. Further, the NS2DS (Park et al., 2020) is used as the experimental model, and Park et al. (2020) is

Dimension	4	3	2	1
Accuracy	Uses sentence structure, vocabulary, and grammar correctly without errors	Uses sentence structure, vocabulary, and grammar correctly with few errors	Uses sentence structure, vocabulary, and grammar correctly with several errors	Uses sentence structure, vocabulary, and grammar correctly (many errors)
Communication	Communicates thoughts and be understood without errors	Communicates thoughts and be understood with few errors.	Communicates thoughts and be understood with several errors.	Not able to communicate thoughts or be understood
Fluency	Communicates clearly and smoothly	Communicates clearly and smoothly with a little hesitation	Able to communicate with some prompts	Not able to communicate clearly or smoothly
Comprehension	Understands and always responds appropriately	Understands most verbal cues and mostly responding appropriately	Understands some verbal cues and sometimes requiring prompts	Not able to understand verbal cues or to respond
Maturity of the language	Includes details beyond the minimum requirements (word-choice/expressions/gestures)	Includes details beyond the minimum requirements	Includes minimal or no details beyond the minimum requirements	Not able to utilize the language well

Table 10: Five dimensions of English-speaking performance evaluation.

referred to for the experimental results.

E Human Evaluation Measures

We ask the following additional questions to the evaluators to assess the quality of the model-generated utterances.

- **Fluency:** Fluency of the dialogue on a scale from 1 to 5, where 1 is “not fluent at all”, 5 is “extremely fluent”, and 3 is “fine”.
- **Situation Consistency:** Situation consistency on a scale of from 1 to 5, where 1 is “not maintained at all”, 5 is “well maintained”, and 3 is “fine”.
- **Engagingness:** Engagingness disregarding fluency from 1 to 5, where 1 is “not engaging at all”, 5 is “extremely engaging”, and 3 is “fine”.

F Details of User Evaluation

F.1 English-speaking Performance

To measure English-speaking performance, we recruited two English teachers who graduated with degrees in English education as evaluators. Following [Chien et al. \(2020\)](#), we modified the methodology to measure five dimensions: Accuracy, communication, fluency, comprehension, and maturity of language. Each dimension was evaluated on a scale of 1 to 4, with 1 being the lowest and 4 being the highest. Table 10 provides a description of each dimension. The inter-rater reliability for the evaluators on the English speaking ability had a Cohen’s kappa value of 0.58 ($p < 0.001$).

F.2 English Grammar Test

For the Grammar evaluation, we use multiple-choice questions to evaluate the participants’ knowledge of English grammar rules. The questions cover a range of topics, such as verb tenses, prepositions, and articles. We administer a grammar test consisting of 20 items for both the pre-test and post-test.

F.3 English Learning Anxiety Scale

The measure for English learning anxiety, which consists of 33 items, was adapted from the Foreign Language Classroom Anxiety Scale ([Horwitz et al., 1986](#)) with a 5-point Likert-type rating, such as “I never feel quite sure of myself when I am speaking in my foreign language class.” A higher rating indicates higher English learning anxiety. The Cronbach’s alpha of the measure was 0.95.

G PEEP-Talk Satisfaction Surveys

Question
Are you satisfied with the overall performance of PEEP-Talk?
Do you think situational dialogue is helpful for conversational learning?
Are you satisfied with the result of the context detector module?
Are you satisfied with the performance of the grammar error correction module?
Do you think the chatbot’s conversation is natural and contextual?

Table 11: Questions of PEEP-Talk satisfaction survey.

We conduct user satisfaction surveys on our platform. Five questions are selected to ask users to

evaluate on a Likert scale(Likert, 1932) of “satisfied”, “neither agree nor disagree”, and “dissatisfied”. The survey questionnaires are listed in Table 11. We show the results of the survey in Figure 8.

As a result of the survey, most people express satisfaction with the performance of PEEP-Talk, and more than 70% of the respondents answer that the utterances of the chatbot are natural and contextual. Also, 70% to 80% of respondents answer that situational dialogue is educationally beneficial for conversational English, and grammar correction is helpful. However, the satisfaction of the CD module is 46%, which is relatively low. We speculate that this is due to the CD module switching topics, even if the user wants to talk more about a particular situation.

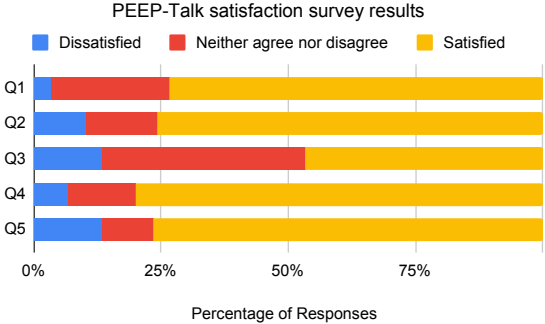


Figure 8: Results on the PEEP-Talk satisfaction survey

G.1 Response Comparison

Table 12 presents a qualitative analysis of the performance of PEEP-Talk in comparison with DailyDialog, Andy, and Speak Now. In each situation, PEEP-Talk provided responses that were contextually appropriate and reflected a natural conversation flow, indicating its potential to be an effective tool for real-world English education. In contrast, the responses generated by the other chatbots sometimes appeared to be unrelated to the topic or situation, suggesting that they may not be suitable for use in English education.

Situation	PEEP-Talk	DailyDialog	Andy	Speak Now
Bookstore There are so many new books this month.	I know, I am looking for a good Korean novel these days.	I know, I'm just impatient.	Do you like to travel?	I'm Okay.
Requesting packaging Can I get the leftovers to go?	You can, but we don't have a container for soup dishes.	I'm sorry, I don't have any leftovers.	Sure, will you write me later?	I'm looking at you
At the art museum Which way should I go to see the special exhibition hall?	If you go to the right, you'll see the entrance of the exhibition hall.	I think you have to go to the main hall.	Is it a good place?	Oh, You hurt my feelings
Customer service I ordered sweet potatoes to go on a diet, but there were some rotten ones included, can I get them exchanged?	I'm sorry, but you can't exchange or refund food that has already been sent.	I don't think you can exchange them.	My name is Andy. What is your name?	I'm okay.
Returning I bought a cell phone case and it's not pretty so I want to get a refund.	I am sorry but cell phone cases are non-refundable.	I bought a case and it's not pretty so I want to get a refund. english haiku bot	Got it	Oh, You hurt my feelings
Find product I'm looking for a lotion for dry skin, is there a cosmetic that is not too sticky?	If you want to use it as a base, I recommend this lotion for oily skin.	I use moisturizer.	I see	Glad to hear that.
Travel agency How much is a cheap flight from Incheon to New York airport?	I heard it is about 80 dollars if you depart from here.	I think it's about the same price as a flight to the US from the UK.	Got it	Let's keep going.
Clothing store I want to buy these clothes, can I try it on first?	You can try it on in the change room, were you not able to find it?	I don't think you can buy them, but you can buy the shirt.	To buy these clothes is a great goal.	What color is your hair?
Job interview Please tell me about your previous job experiences if any.	I worked in a bookstore as a part-time job and had a lot of free time.	I have a few. I've been in the service for a few years.	As you know my name is Andy and I want to help you in learning English.	Let's keep going.
IT/computer Is this the IT department? The program isn't working, please check it.	Yes, which program isn't working?	I'm not sure what you mean.	Thank you! I'm so happy to hear that, I love talking to you too	I'm Okay.
Daily Conversation It seems like you can't go home often ever since you moved out from home.	I can't go home often because I have too many friends who are also students.	I've been home for a few months now.	Got it	Thanks. Have a nice day.
Inquiring about prices Are there clothes on sale now?	Yes, we have sales clothes on this side separately.	I don't think so. I think they are just a limited time thing.	I see	Great. Man.
Duty-free shop I was told that the cigarettes are really cheap if bought at duty-free shops.	I heard that too, but isn't it just cigarettes?	I've never seen a duty free shop that has them.	Okay	I don't understanding what you are saying.

Table 12: Comparison of the utterances generated by PEEP-Talk, DailyDialog Andy, and Speak Now.

H Example Conversations

We show a sample conversations of Andy, Speak Now, and PEEP-Talk.

H.1 Conversations between Humans and PEEP-Talk (Ours)

PEEP-Talk (Ours) Conversation 1	
Situation:	At the cosmetic store
Human:	<i>What are some products on sale today?</i>
PEEP-Talk:	<i>We also have a 10 percent discount on lotion and toner.</i>
Human:	<i>This is the product I am using, and do you still sell it?</i>
PEEP-Talk:	<i>We do, but we only have a small amount left.</i>
Human:	<i>I'd like to buy a set of cosmetics.</i>
PEEP-Talk:	<i>I'll go get it for you.</i>
Human:	<i>Thank you. Have a nice day</i>

PEEP-Talk (Ours) Conversation 2	
Situation:	Evaluating food taste
Human:	<i>I heard that salmon salad is really famous at this restaurant.</i>
PEEP-Talk:	<i>Really? I love salmon salad . I also heard that the Stir-Fried mushrooms are also very famous</i>
Human:	<i>Then, shall we order a salmon salad and a pizza and share them?</i>
PEEP-Talk:	<i>I don't think I can eat Pizza alone. Let's order some more and share them.</i>
Human:	<i>Today the weather is good.</i>
PEEP-Talk:	<i>You are right. The temperature is good and the wind is good.</i>
Human:	<i>Good.</i>
PEEP-Talk:	<i>--Situation switch required--</i>
PEEP-Talk:	<i>Why don't we speak another situation? please, click the different situation!</i>

PEEP-Talk (Ours) Conversation 3	
Situation:	Situations diagnosed in the hospital
Human:	<i>I have a headache</i>
PEEP-Talk:	<i>When did the headache start?</i>
Human:	<i>I started having symptoms last week.</i>
PEEP-Talk:	<i>It seems you have a persistent headache.</i>
Human:	<i>And I have chills and fever.</i>
PEEP-Talk:	<i>I think we need to do a detailed examination.</i>
Human:	<i>Yes, thank you. Where is it going to take a place?</i>
PEEP-Talk:	<i>I will go to the central nervous system and tell you the location.</i>
Human:	<i>Okay. Thank you</i>
PEEP-Talk:	<i>Here is a paper with a magnet attached to it . please follow me.</i>

PEEP-Talk (Ours) Conversation 4	
Situation:	School life
Human:	<i>Professor, when will you hand out the midterm results we took last week?</i>
PEEP-Talk:	<i>I am writing them now . do you want them by any chance?</i>
Human:	<i>Could you please tell us the highest and lowest scores in advance?</i>
PEEP-Talk:	<i>The students usually get an 85 on the Midterms and the lowest is 70 points.</i>
Human:	<i>Okay. Thank you</i>
PEEP-Talk:	<i>You're welcome . Let me know if you have any questions.</i>
Human:	<i>How do we decide the groups? Will you be randomly assigning them?</i>
PEEP-Talk:	<i>I will assign them based on how many friends are in the class.</i>
Human:	<i>How many people are assigned per group?</i>
PEEP-Talk:	<i>I will assign 5 people to a group of 15.</i>

Table 13: Example conversations between humans and our proposed PEEP-Talk. It is able to not only converse in a certain real-world situation but also can switch the situation or topic when the users input the utterance predicted as it is out of the situation (Conversation 2).

H.2 Conversations between Humans and Andy

Andy Conversation 1	
Andy:	<i>Hello.</i>
Human:	<i>Good morning.</i>
Andy:	<i>My name is Andy. What is your name?</i>
Human:	<i>My name is Jun.</i>
Andy:	<i>That's great! I am happy for you.</i>
Human:	<i>What are you doing?</i>
Andy:	<i>I was doing some exercises. I always feel sleepy and sad without physical activity.</i>
Human:	<i>Me too. I have to work out in the morning and evening.</i>
Andy:	<i>Interesting.</i>

Andy Conversation 2	
Andy:	<i>Where are you from?</i>
Human:	<i>I came from Canada.</i>
Andy:	<i>Is it a good place?</i>
Human:	<i>I like a fencing.</i>
Andy:	<i>Oh, I want to visit it now</i>
Human:	<i>Come here. Anytime.</i>
Andy:	<i>Nice. Do you have brothers or sisters?</i>
Human:	<i>Yes.</i>
Andy:	<i>And I have a little sister. She studies modern art in Rome. She is very creative person. I'm sure she will become a famous artist.</i>

Andy Conversation 3	
Andy:	<i>Hi, Jun. What's up</i>
Human:	<i>How are you?</i>
Andy:	<i>I'm doing great, thank you my friend.</i>
Human:	<i>What do you think of the Lakers this season?</i>
Andy:	<i>It's very sunny!</i>
Human:	<i>This team is too old. We need new blood.</i>
Andy:	<i>Yes, I'm completely fool</i>
Human:	<i>All our family members always have been supporting the same team.</i>
Andy:	<i>Nice. Do you know any jokes?</i>
Andy:	<i>No</i>

Andy Conversation 4	
Andy:	<i>How do you think cities will change in the future?</i>
Human:	<i>Nothing much, just thinking about the good old days when I was a kid. How about you?</i>
Andy:	<i>I'm doing great, thank you!</i>
Human:	<i>That's great to hear! I hope you have a great rest of your day! :)</i>
Andy:	<i>I understand. Of all the people in history that have reached 65 years of age, half of them are living right now.</i>
Human:	<i>Wow, that's a lot of people to live with. Do you have any siblings?</i>
Andy:	<i>Yeah, amazing. What's your father like?</i>
Human:	<i>He's a great man. He's always been there for me when I needed him the most.</i>
Andy:	<i>Got it. Can you tell about your parents?</i>

Table 14: Example conversations between humans and Andy. The responses of Andy are restricted and cannot cover a variety of topics.

H.3 Conversations between Humans and Speak Now

Speak Now Conversation 1	
Speak Now:	<i>What's up. Did you have breakfast?</i>
Human:	<i>No, I haven't yet.</i>
Speak Now:	<i>Oh, I'm sorry to hear that. Do you like seafood.</i>
Human:	<i>Yes. I like shrimp.</i>
Speak Now:	<i>How we should go to Busan someday to eat seafood. Do you like meat?</i>
Human:	<i>Yes, I like it.</i>
Speak Now:	<i>Good. We should eat meat someday. What's your favorite food?</i>
Human:	<i>My favorite food is sushi</i>
Speak Now:	<i>How we have same taste. Do you like breads?</i>

Speak Now Conversation 2	
Speak Now:	<i>What's up. Did you have breakfast?</i>
Human:	<i>Yes, I ate 15min ago.</i>
Speak Now:	<i>I'm doing okay man. What would you like to talk about today?</i>
Human:	<i>Let's talk about cars</i>
Speak Now:	<i>No. I'm not sure about that. We Should talk about something else.</i>
Human:	<i>Why don't we talk about colors?</i>
Speak Now:	<i>No. I can not understand. Have you ever went study aboard?</i>
Human:	<i>No. I have not. But, I want to go further day.</i>
Speak Now:	<i>Oh. Should Have you ever been to England?</i>

Speak Now Conversation 3	
Speak Now:	<i>Did you Sleep well, last night?</i>
Human:	<i>I didn't sleep very well</i>
Speak Now:	<i>Oh. I sorry to hear that. Do you like drinking alone?</i>
Human:	<i>No, I did not like alcohol</i>
Speak Now:	<i>What's your favorite alcoholic drink?</i>
Human:	<i>I did not like alcohol</i>
Speak Now:	<i>Have you invested in Bitcoin before?</i>
Human:	<i>No, I have not</i>
Speak Now:	<i>A lot of people lost money for that. What would you like to talk about today?</i>
Human:	<i>I want to talk about soccer;</i>
Speak Now:	<i>I don't like topic. Let's try different one</i>

Speak Now Conversation 4	
Speak Now:	<i>Hey Dude did you sleep well last night</i>
Human:	<i>I did sleep well</i>
Speak Now:	<i>Do you like your Boss?</i>
Human:	<i>No I don't like my boss</i>
Speak Now:	<i>I understand you I think most people do. Are you satisfied your job?</i>
Human:	<i>No I do not satisfy my job .</i>
Speak Now:	<i>I understand you it can always be fun. Which country are you from?</i>
Human:	<i>I came from Korea</i>
Speak Now:	<i>How you live in a good place Where do you live?</i>

Table 15: Example conversations between humans and Speak Now. We used free conversation mode of Speak Now. As Speak Now uses hand-crafted rules and pattern matching when making the utterances, the utterances are limited to its own pre-built templates.

OpenTIPE: An Open-source Translation Framework for Interactive Post-Editing Research

Fabian Landwehr and Thomas Steinmann and Laura Mascarell

ETH Zurich

{fabian.landwehr,thomas.steinmann,lmascarell}@inf.ethz.ch

Abstract

Despite the latest improvements on machine translation, professional translators still must review and post-edit the automatic output to ensure high-quality translations. The research on automating this process lacks an interactive post-editing environment implemented for this purpose; therefore, current approaches do not consider the human interactions that occur in real post-editing scenarios. To address this issue, we present OpenTIPE, a flexible and extensible framework that aims at supporting research on interactive post-editing. Specifically, the interactive environment of OpenTIPE allows researchers to explore human-centered approaches for the post-editing task. We release the OpenTIPE source code¹ and showcase its main functionalities with a demonstration video² and an online live demo.³

1 Introduction

Recent advances in Machine Translation (MT) and the adoption of neural architectures (Bentivogli et al., 2016) led to significant improvements in translation quality aspects such as fluency or adequacy (Bentivogli et al., 2016; Bojar et al., 2017). Despite these advancements, MT is not yet on a par with human performance (Läubli et al., 2018; Toral, 2020) and human post-editors need to edit the MT output to obtain high-quality translations that also adapt to a specific domain and style.

To reduce the manual efforts, the research community proposed to automate this process and implemented Automatic Post-Editing (APE) models that automatically learn post-editing rules from revised translations (do Carmo et al., 2020). The use of these models is specially indicated in environments where the MT system that generates the translations is not accessible for retraining (Chatterjee et al., 2015). To date, the automatic corrections

generated by the state-of-the-art APE models still require proofreading, so there is no solution that fully automates the translation process. In fact, the post-editing task remains mostly manual in production workflows.

Instead of fully automating the post-editing task, Escribe and Mitkov (2021) suggest that post-editing would greatly benefit from human-centered approaches that leverage the post-editor’s corrections and their interactions with the translation interface. For example, APE models could improve over time by incrementally learning from human corrections (Chatterjee et al., 2017).

While human-computer interaction has been explored in MT with the help of translation frameworks such as CASMACAT (Alabau et al., 2013), there is no such interactive environment designed for the post-editing task (do Carmo et al., 2020; Escribe and Mitkov, 2021). Therefore, current research in Interactive Post-Editing (IPE) is limited to simulate the human interaction by feeding pre-existing corrections to the post-editing process sequentially (Ortiz-Martínez and Casacuberta, 2014; Chatterjee et al., 2017; Negri et al., 2018a). Additionally, human corrections are scarce and these approaches often rely on synthetic post-edited datasets such as eSCAPE (Negri et al., 2018b). Although these artificial settings enabled valuable research in this field, they lack the human intervention as in real-world post-editing scenarios.

In this paper, we present OpenTIPE, an open-source framework that enables research on post-editing in an interactive environment. In particular, OpenTIPE implements the following main features:

- **Easy-to-use interface** that allows users to automatically translate a text and post-edit it. To support the user during the post-editing process, the tool provides **automatic post-editing suggestions** from an APE model, which can be directly applied to the revised translation (see Section 3.1).

¹Link to GitHub repository.

²https://youtu.be/G3Hb8_hnKIk

³<https://www.opentipe-demo.com>

- **Collection of human data**, such as user corrections and post-editing feedback (e.g. whether the automatic suggestions were applied). The collected data is a valuable resource to incrementally improve APE models in a continuous feedback loop (Section 4).
- **Logging of post-editing activity**, such as the user inactivity, and the time at the start and end of the post-editing task. The implemented logging allows researchers to measure post-editing efforts and evaluate the post-editor experience on different settings (Section 4).
- **Modular and extensible** microservice architecture using Docker containers,⁴ which facilitates the extension and implementation of additional services (e.g. translation or APE models) and features (e.g. new logging activity or user interface design). Section 3 describes the OpenTIPE architecture in detail.

To the best of our knowledge, this is the first interactive environment designed to facilitate research on IPE. We hope that OpenTIPE fosters further research in this field that can be applied to improve the overall post-editing experience.

2 Related Work

Most of the related work focuses on implementing APE approaches to automate the post-editing task. However, these approaches do not enhance human-computer interaction. In fact, human-centered approaches have been only explored in MT settings. In this section, we first summarize the work on APE and their online approaches, which simulate the post-editor behaviour by implementing continuous streams of corrections. We then describe the research on interactive MT, which is the most in-line with this work.

Automatic Post-editing The annual WMT conference⁵ has been hosting shared tasks on APE since 2015, going through both statistical and neural MT eras (Junczys-Dowmunt, 2018; do Carmo et al., 2020). An important finding is that the performance of APE models are highly influenced by the quality of the MT output, hence improving neural MT translations is particularly challenging for these models. Additionally, automatic metrics such as TER (Snover et al., 2006) and BLEU (Papineni

et al., 2002) cannot always reflect the improvements of the APE models, and researchers need to conduct manual evaluations to gain insights on the quality of the APE output (Akhbardeh et al., 2021). Nevertheless, APE models are an essential component in translation workflows, where the MT system is used as a black box and its parameters are not available for retraining. In this setting, it would be beneficial to leverage human feedback to gradually improve the performance of the APE model in place.

Online APE To simulate human post-editions, APE models apply online learning methods that feed continuous streams of data to the model. While online methods have been previously adopted in phrase-based APE (Ortiz-Martínez and Casacuberta, 2014; Simard and Foster, 2013; Chatterjee et al., 2017), only Negri et al. (2018a) apply online learning to neural APE. Specifically, Negri et al. (2018a) iterate over a pre-existing dataset of corrections, updating the model parameters on the fly for every instance. Similar works address online learning in neural MT. For example, Kothur et al. (2018) update the model parameters one sentence at a time as in Negri et al. (2018a). In contrast, other approaches avoid updating the model parameters and retrieve sentences with similar contexts from a translation memory during decoding (Gu et al., 2018; Wang et al., 2021).

Interactive MT In professional translation environments, human experts benefit from using computer-assisted translation technologies (i.e. CAT tools). For example, translation memories store previously-approved translations, so they can be reused later on. To further investigate the human-computer interaction in translation workflows, researchers proposed several frameworks for phrase-based MT (e.g. Transtype2 (Esteban et al., 2004) and CASMACAT (Alabau et al., 2013)) and neural MT (Knowles and Koehn, 2016; Santy et al., 2019). However, these technologies are not optimal to investigate human interaction in post-editing, and therefore there is a lack of research in this area. For example, CASMACAT offers alternative translation suggestions that come from an MT system, whereas our work integrates the output of an APE model. Similarly to the prior work in interactive MT, we implement automatic logging strategies to collect user interactions during the post-editing process for further analyses.

⁴<https://www.docker.com>

⁵<https://www.statmt.org/wmt22/>

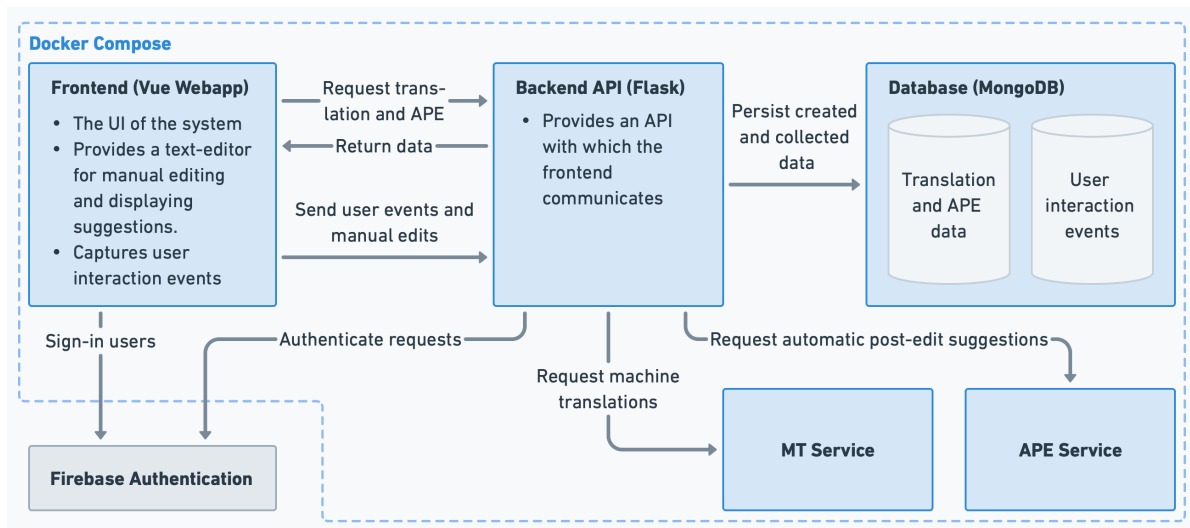


Figure 1: Overview of the OpenTIPE system architecture, which follows a microservice pattern, and the communication between the different components. Each blue box in the diagram represents a separate Docker container.

3 The OpenTIPE Framework

OpenTIPE implements a microservice architecture, consisting of independent services that are orchestrated with Docker Compose.⁶ The advantages of this architecture are twofold. First, it enables separation of concerns. That is, each service implements its own function and runtime environment, reducing code complexity and errors. Second, its flexibility, as services can be easily replaced.

The main components of the OpenTIPE implementation are the frontend, which provides the graphical user interface to translate and post-edit texts, and the backend services: the backend API, MT and APE services, and data storage. Additionally, OpenTIPE supports user authentication with Firebase.^{7,8} Figure 1 illustrates the overall architecture of OpenTIPE and the interaction between the different services. In the following, we describe the technical details of each component.

3.1 Frontend

The frontend implements the user interface of OpenTIPE, which currently consists of two main views: (1) the translation and (2) post-editing view (Figure 2). The translation view allows the user to add the text to translate and select different translation options. In particular, the user can choose among the available source and target languages, and define the translation of specific terms (see

Section 3.4 for more details on the use of lexical constraints). In the post-editing view, the user can edit the automatic translation with the support of the post-editing suggestions from the APE model.

To facilitate its deployment, we adopt a web-based design. More specifically, we use VueJS, a lightweight JavaScript framework, and build the frontend as a single-page application.⁹ This implementation runs entirely in the browser and decouples the backend business logic from the user interface, enhancing separation of concerns. To obtain the translations and the automatic post-editing suggestions, the frontend application communicates with the backend API (see Section 3.2).

An important feature of the user interface is its rich-text editor, which is implemented using the Tiptap framework.¹⁰ The main strength of the Tiptap framework is its extensibility, allowing us to easily customise and add additional features. We write the entire frontend application in TypeScript¹¹ and host it statically using NGINX¹² in its corresponding docker container.

Document-level Post-editing Computer-assisted translation technologies typically organise the translation task in individual sentences. The human translator then addresses the document sentences one at a time, which helps to speed up the translation process. However, prior work reported that errors in current high-quality MT systems are harder

⁶<https://docs.docker.com/compose/>

⁷<https://firebase.google.com/docs/auth>

⁸Authentication can be entirely disabled if necessary.

⁹<https://vuejs.org>

¹⁰<https://tiptap.dev>

¹¹<https://www.typescriptlang.org>

¹²<https://www.nginx.com>

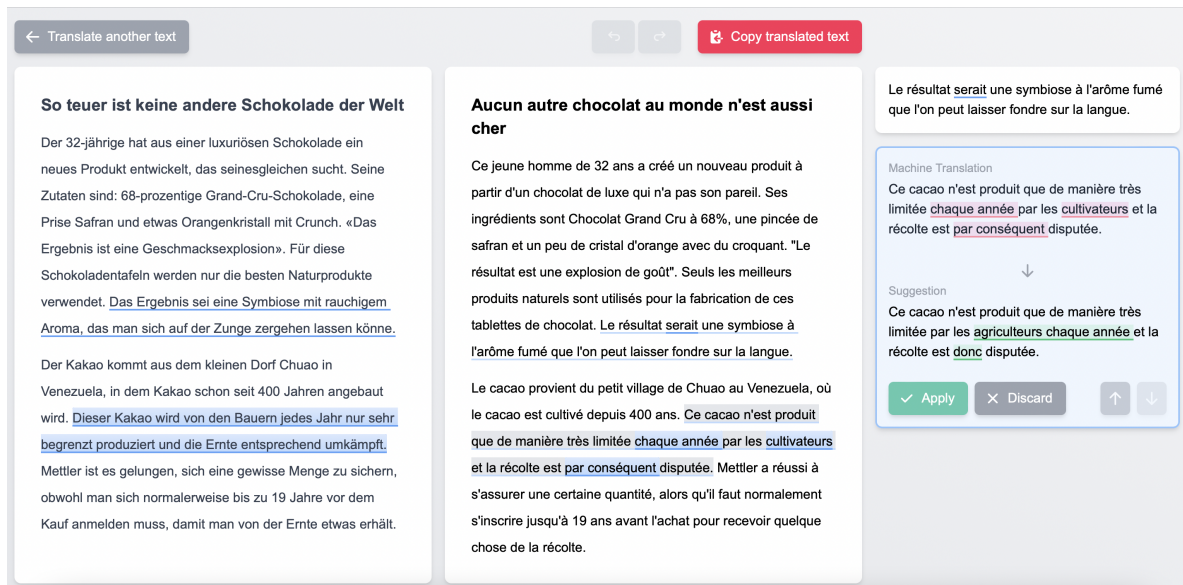


Figure 2: Post-editing view of OpenTipe. The view shows the source text on the left, the post-editing area in the middle, which initially contains the MT output, and the APE suggestions on the right side. The user can discard or apply the suggestions to the revised text. The interface implements highlighting features to identify aligned source-target sentences and the differences between the translation and the corresponding APE suggestions. Users can click on the ‘Copy translated text’ button to save the current status of the revised translation locally.

to spot when focusing on isolated sentences instead of the whole document (Läubli et al., 2018). Additionally, a professional translator, who took part in our observation-based study (see details in Section 5), confirmed us the importance of document-level post-editing. We therefore display the entire document without segmentation in the post-editing view and offer highlighting features, which help the user to identify aligned sentences in the source and the target (see Figure 2). Nevertheless, the backend deals with the source, translation, and post-edited texts as independent sentences,¹³ so it is possible to modify the user interface and display separate pairs of source-target sentences instead.

3.2 Backend API

The backend API acts as bridge between the frontend and the backend services (i.e. the database, the MT and APE models), defining the necessary endpoints to allow them to communicate and share resources. Specifically, it first provides the frontend with the automatic translations and post-editing suggestions from the MT and APE services, respectively (Section 3.4). Second, it stores the resulting post-editing metadata such as human post-edits and user interactions to the database (Section 3.3).

The API is based on Flask, a framework de-

¹³We refer the reader to Section 4 for more details on the format and structure of the data.

signed to build lightweight web applications,¹⁴ and implemented in Python.¹⁵ It is the only backend service that is accessible from the internet. Thus, if authentication is enabled, it connects to Firebase to validate and authenticate the incoming requests.

3.3 Data Storage

The data storage is based on MongoDB, a popular document-oriented NoSQL database.¹⁶ The main strengths of MongoDB are its high performance and flexibility. In contrast to relational databases, MongoDB can support different data structures. Therefore, researchers can easily extend and modify the current implementation to fulfill their needs.

We divide the storage of the collected data, that is, the post-editing metadata and the logging of the user interactions, into two logical databases. Section 4 describes the data collection and its representation in more detail.

3.4 Translation and Post-editing Models

The OpenTipe framework uses MT and APE models to obtain automatic translations and post-editing suggestions, respectively. We build these models in independent Docker containers, so they can be easily replaced. The current implementation of the MT

¹⁴<https://flask.palletsprojects.com/en/2.2.x/>

¹⁵<https://www.python.org>

¹⁶<https://www.mongodb.com>

mtText	apeText	hpeText	apeAccepted
The Bremen town musicians	Town musicians of Bremen	Town musicians of Bremen	true
The Bremen town musicians	The Bremen town musicians	Town musicians of Bremen	false
Town musicians of Bremen	Town musicians of Bremen	Town musicians of Bremen	false
The Bremen town musicians	Town musicians of Bremen	The town musicians of Bremen	false
The Bremen town musicians	Town musicians of Bremen	The Bremen town musicians	false

Table 1: Examples of different values of the *textSegments* properties for a single sentence object (*srcText*: ‘Die Bremer Stadtmusikanten’). If there is no automatic suggestion (i.e. second and third rows), *apeText* contains the *mtText*. The value of *hpeText* is the final version of the sentence even if there is no correction.

service supports the use of the DeepL API¹⁷ and Huggingface¹⁸ or Fairseq Neural MT models.¹⁹

APE and Lexical Constraints In this post-editing environment, we consider the MT model as a black box and the improvements should be applied to the APE model. As an example, we release a simple APE implementation, an encoder-decoder architecture as in [Correia and Martins \(2019\)](#). In contrast to multi-source architectures, [Correia and Martins \(2019\)](#) use a single encoder whose input is the concatenated source and MT translation.²⁰

Since post-editors are often required to use translation dictionaries, we extend the APE implementation to allow lexical constraints. That is, we can enforce the APE model to use specific translations for particular terms. To do so, we follow the approach described in [Bergmanis and Pinnis \(2021\)](#), which also handles the generation of the correct morphological inflection in the APE output.²¹ This is specially important when translating into an inflected language, such as French. The approach augments the APE training data with the lemma of nouns and verbs such that the model learns to *copy and inflect*. For example, given the source text ‘the improvement’, we would augment it with the noun lemma in the target language (e.g. ‘the improvement *retouche*’). As in [Bergmanis and Pinnis \(2021\)](#), we use the pre-trained Stanza models for lemmatization and part-of-speech tagging.²² To define the translation constraints, the user can provide them in a file or introduce manual entries in the dictionary view of the user interface. During inference, we only augment those terms in the source for which the user specified a translation.

¹⁷<https://www.deepl.com/docs-api>

¹⁸<https://huggingface.co/models>

¹⁹<https://github.com/facebookresearch/fairseq/blob/main/examples/translation/README.md>

²⁰<https://github.com/deep-spin/OpenNMT-APE>

²¹Jon et al. (2021) propose a similar approach to enforce lexical constraints and generate the corresponding inflection.

²²<https://github.com/stanfordnlp/stanza>

key	description
<i>srcLang</i>	Language code of the source text.
<i>trgLang</i>	Language code of the target text.
<i>userDict</i>	Array of the manual translation entries defined by the user in the user interface.
<i>selectedDicts</i>	Array of the predefined dictionaries selected by the user.
<i>textSegments</i>	Array of sentence objects. Each sentence object contains the corresponding values for the source sentence (<i>srcText</i>), MT translation (<i>mtText</i>), automatic post-editing suggestion (<i>apeText</i>), revised version (<i>hpeText</i>), and a boolean indicating whether the automatic suggestion was accepted (<i>apeAccepted</i>).

Table 2: Description of the JSON object properties that define a post-edited document. See examples of sentence objects from *textSegments* in Table 1.

4 Data Collection and Representation

Human post-edited data is a valuable resource to improve APE models. However, this is a scarce resource and researchers are often dependent on synthetic datasets. Therefore, the collection of human data is an important aspect of this IPE environment. Researchers can then leverage the data to implement human-in-the-loop approaches and assess the performance of different APE settings. Our implementation of OpenTIPE collects (1) human post-edited translations and (2) user interactions with the post-editing environment and it stores them as JSON objects in the MongoDB database (Section 3.3). The rest of this section describes the representation of these data in more detail.

Human Post-edited Translations We collect the human corrections together with additional relevant information, such as the corresponding source, MT output and use of translation dictionaries. OpenTIPE deals with the data at sentence-level, aligning sentence quartets of source, MT translation, APE suggestion, and proofread version. The

data is represented as a JSON object that defines the properties listed in Table 2. OpenTIPE captures and stores this data in the database when the user triggers the end of a revised version. That is, when the user copies the post-edited text using the copy keyboard shortcut or clicks on the ‘Copy translated text’ button of the user interface (see Figure 2). Note that different revisions of the same source text can be stored at different times.

User Interaction Logging We also record the user interactions with OpenTIPE in our database. This data can be used to evaluate different APE settings and the user experience with the editor. We currently log five types of events: *IdleEvent*, *ActiveEvent*, *AcceptEvent*, *RejectEvent*, and *CopyEvent* together with the timestamp and user identifier.²³ The pair of events *IdleEvent* and *ActiveEvent* indicate the time intervals with user activity and inactivity. Specifically, we record an *IdleEvent* when the user does not interact with the interface for a minute and an *ActiveEvent* with any interaction after being idle (e.g. mouse click, scrolling). The event types *AcceptEvent* and *RejectEvent* are triggered when the user applies or discards automatic suggestions, respectively. Finally, *CopyEvent* indicates that the user copied a post-editing revision locally.

5 Usability Study

We perform a user study to assess the usability of the OpenTIPE user interface. In particular, we conduct a controlled observation with a professional translator and a survey-based assessment with eight non-professional translators. The latter are academics between 21 and 30 years old (62.5% are male and 37.5% female), who indicated that they frequently use translation services, such as DeepL²⁴ and Google Translate.²⁵ While the observation-based setting allows us to get insights on the interactions of an expert with the tool, the survey-based assessment gives us a general subjective view of the user interface usability.

In both settings, all participants saw the interface of OpenTIPE for the first time during the study. We start the study explaining its purpose to the participants. In the observation-based setting, the professional translator is aware that he is being observed

during the process.²⁶ We then provide them with a text to translate and the following instructions:

1. Translate the provided text using OpenTIPE.
2. Improve the automatic translation. For example, (a) apply automatic suggestions where needed or (b) rephrase the first sentence and split it in two sentences.
3. Save the final translation locally.

Furthermore, we ask the participants of the survey-based setting to fill in a questionnaire. The questionnaire consists of a set of questions as defined in the System Usability Scale (SUS) (Brooke, 1996) and three additional qualitative questions about what they liked the most and what features they think are missing or could be improved.

The OpenTIPE user interface obtained an average SUS score of 90, being 85 the lowest among the participants.²⁷ These results indicate that all participants evaluated the interface as excellent (Bangor et al., 2008). This is also confirmed with the answers to the qualitative questions. In fact, most of them stated that what they liked the most about the interface was its simplicity. Similarly, we observed that the professional translator used the interface as expected and could perform all tasks effortlessly.

6 Conclusion

We presented OpenTIPE, the first interactive framework that aims at supporting the research of human-centered approaches for post-editing. In contrast to research in machine translation, human-computer interaction has been only simulated for the post-editing task, since there was no interactive environment available for this purpose. OpenTIPE follows a microservice architecture such that it can be easily extended and adapted to other models or features. Additionally, it collects human post-editing data and the user interactions with the interface. These data are key to implement human-in-the-loop approaches that learn from human corrections over time. We expect this work to foster future research on interactive approaches that enhance the performance of the post-editing process. We are excited to explore this direction in future work.

²³The logging can be easily extended with new event types.

²⁴<https://www.deepl.com/translator>

²⁵<https://translate.google.com>

²⁶Two authors of this paper participated as observers.

²⁷SUS scores have a range of 0 to 100 and a score over 68 is considered above average.

Ethics Statement

Usability Study We recruited the participants for our usability study on a voluntary basis and informed them of the goals and scope. Furthermore, we collected the data anonymously, such that no conclusion can be drawn about any participant. The usability study obtained the ethical approval (EK-2023-N-35) from the Ethics Commission of ETH Zurich university.

Translation and Post-editing Models We do not expect additional ethical concerns besides the already documented on natural language generator systems (Smiley et al., 2017; Kreps et al., 2022).

Potential Misuse Users could write undesired text (e.g. hateful or offensive comments) as post-edited text. As a result, the stored data could be used to train a model to generate texts that replicate this harmful behaviour. To mitigate this issue, we strongly recommend to activate the user authentication, so the framework is only accessible to trustworthy users. Additionally, researchers should periodically verify the data to filter those instances either manually or automatically, using a model to identify hallucinations in the text as in Su et al. (2022). Since bias can be present in the APE output, human-in-the-loop approaches can amplify this bias if the users heavily rely on the APE suggestions. Therefore, researchers should also debias the data regularly, for example, using existing tools such as AdaTest (Ribeiro and Lundberg, 2022).

Acknowledgements

This project is supported by Ringier, TX Group, NZZ, SRG, VSM, viscom, and the ETH Zurich Foundation.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. [Findings of the 2021 conference](#)

[on machine translation \(WMT21\)](#). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González-Rubio, Philipp Koehn, Luis A. Leiva, Bartolomé Mesa-Lao, Daniel Ortiz, Herve Saint-Amand, Germán Sanchis-Trilles, and Chara Tsoukala. 2013. [CASMACAT: An open source workbench for advanced computer aided translation](#). *Prague Bulletin of Mathematical Linguistics*, 100:101–112.

Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. [An empirical evaluation of the system usability scale](#). *International Journal of Human–Computer Interaction*, 24(6):574–594.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. [Neural versus phrase-based machine translation quality: a case study](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Toms Bergmanis and Mārcis Pinnis. 2021. [Facilitating terminology translation with target lemma annotations](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3105–3111, Online. Association for Computational Linguistics.

Ondřej Bojar, Jindřich Helcl, Tom Kocmi, Jindřich Libovický, and Tomáš Musil. 2017. [Results of the WMT17 neural MT training task](#). In *Proceedings of the Second Conference on Machine Translation*, pages 525–533, Copenhagen, Denmark. Association for Computational Linguistics.

John Brooke. 1996. [SUS: A quick and dirty usability scale](#), volume 189, pages 4–7. Taylor & Francis.

Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri, and Marco Turchi. 2017. [Online automatic post-editing for MT in a multi-domain translation environment](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 525–535, Valencia, Spain. Association for Computational Linguistics.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015. [Exploring the planet of the APEs: a comparative study of state-of-the-art methods for MT automatic post-editing](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 156–161, Beijing, China. Association for Computational Linguistics.

- Gonçalo M. Correia and André F. T. Martins. 2019. [A simple and effective approach to automatic post-editing with transfer learning](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy. Association for Computational Linguistics.
- Félix do Carmo, Dimitar Shterionov, Joss Moorkens, Joachim Wagner, Murhaf Hossari, Eric Paquin, Dag Schmidtke, Declan Groves, and Andy Way. 2020. [A review of the state-of-the-art in automatic post-editing](#). *Machine Translation*, pages 1–43.
- Marie Escribe and Ruslan Mitkov. 2021. [Interactive models for post-editing](#). In *Proceedings of the Translation and Interpreting Technology Online Conference*, pages 167–173, Held Online. INCOMA Ltd.
- José Esteban, José Lorenzo, Antonio S. Valderrábanos, and Guy Lapalme. 2004. [TransType2 - an innovative computer-assisted translation system](#). In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 94–97, Barcelona, Spain. Association for Computational Linguistics.
- Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. [Search engine guided neural machine translation](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Josef Jon, João Paulo Aires, Dusan Varis, and Ondřej Bojar. 2021. [End-to-end lexically constrained machine translation for morphologically rich languages](#). 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 4019–4033, Online. Association for Computational Linguistics.
- Marcin Junczys-Dowmunt. 2018. [Are we experiencing the golden age of automatic post-editing?](#) In *Proceedings of the AMTA 2018 Workshop on Translation Quality Estimation and Automatic Post-Editing*, pages 144–206, Boston, MA. Association for Machine Translation in the Americas.
- Rebecca Knowles and Philipp Koehn. 2016. [Neural interactive translation prediction](#). In *Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track*, pages 107–120, Austin, TX, USA. The Association for Machine Translation in the Americas.
- Sachith Sri Ram Kothur, Rebecca Knowles, and Philipp Koehn. 2018. [Document-level adaptation for neural machine translation](#). In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 64–73, Melbourne, Australia. Association for Computational Linguistics.
- Sarah Kreps, R. Miles McCain, and Miles Brundage. 2022. [All the news that's fit to fabricate: AI-generated text as a tool of media misinformation](#). *Journal of Experimental Political Science*, 9(1):104–117.
- Samuel Lüubli, Rico Sennrich, and Martin Volk. 2018. [Has machine translation achieved human parity? a case for document-level evaluation](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. Association for Computational Linguistics.
- Matteo Negri, Marco Turchi, Nicola Bertoldi, and Marcello Federico. 2018a. [Online neural automatic post-editing for neural machine translation](#). In *Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*.
- Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018b. [ESCAPE: a large-scale synthetic corpus for automatic post-editing](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Daniel Ortiz-Martínez and Francisco Casacuberta. 2014. [The new thot toolkit for fully-automatic and interactive statistical machine translation](#). In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 45–48, Gothenburg, Sweden. Association for Computational Linguistics.
- 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*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Marco Tulio Ribeiro and Scott Lundberg. 2022. [Adaptive testing and debugging of NLP models](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, Dublin, Ireland. Association for Computational Linguistics.
- Sebastin Santy, Sandipan Dandapat, Monojit Choudhury, and Kalika Bali. 2019. [INMT: Interactive neural machine translation 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): System Demonstrations*, pages 103–108, Hong Kong, China. Association for Computational Linguistics.
- Michel Simard and George Foster. 2013. [PEPr: Post-edit propagation using phrase-based statistical machine translation](#). In *Proceedings of Machine Translation Summit XIV: Papers*, Nice, France.
- Charese Smiley, Frank Schilder, Vassilis Plachouras, and Jochen L. Leidner. 2017. [Say the right thing right: Ethics issues in natural language generation systems](#). In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages

103–108, Valencia, Spain. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. [A study of translation edit rate with targeted human annotation](#). In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Dan Su, Xiaoguang Li, Jindi Zhang, Lifeng Shang, Xin Jiang, Qun Liu, and Pascale Fung. 2022. [Read before generate! faithful long form question answering with machine reading](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 744–756, Dublin, Ireland. Association for Computational Linguistics.

Antonio Toral. 2020. [Reassessing claims of human parity and super-human performance in machine translation at WMT 2019](#). In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 185–194, Lisboa, Portugal. European Association for Machine Translation.

Dongqi Wang, Haoran Wei, Zhirui Zhang, Shujian Huang, Jun Xie, Weihua Luo, and Jiajun Chen. 2021. [Non-parametric online learning from human feedback for neural machine translation](#). *CoRR*, abs/2109.11136.

TencentPretrain: A Scalable and Flexible Toolkit for Pre-training Models of Different Modalities

Zhe Zhao^{1,*}, Yudong Li², Cheng Hou¹, Jing Zhao¹, Rong Tian¹, Weijie Liu¹, Yiren Chen¹, Ningyuan Sun¹, Haoyan Liu¹, Weiquan Mao¹, Han Guo¹, Weigang Guo¹, Taiqiang Wu¹, Tao Zhu¹, Wenhong Shi³, Chen Chen¹, Shan Huang¹, Sihong Chen¹, Liqun Liu¹, Feifei Li¹, Xiaoshuai Chen¹, Xingwu Sun¹, Zhanhui Kang¹, Xiaoyong Du³, Linlin Shen², Kimmo Yan¹

¹ Tencent AI Lab

² School of Computer Science and Software Engineering, Shenzhen University

³ School of Information and DEKE, MOE, Renmin University of China

Abstract

Recently, the success of pre-training in text domain has been fully extended to vision, audio, and cross-modal scenarios. The proposed pre-training models of different modalities are showing a rising trend of homogeneity in their model structures, which brings the opportunity to implement different pre-training models within a uniform framework. In this paper, we present TencentPretrain, a toolkit supporting pre-training models of different modalities. The core feature of TencentPretrain is the modular design. The toolkit uniformly divides pre-training models into 5 components: *embedding*, *encoder*, *target embedding*, *decoder*, and *target*. As almost all of common modules are provided in each component, users can choose the desired modules from different components to build a complete pre-training model. The modular design enables users to efficiently reproduce existing pre-training models or build brand-new one. We test the toolkit on text, vision, and audio benchmarks and show that it can match the performance of the original implementations.

1 Introduction

Pre-training on large-scale data and then fine-tuning on downstream tasks has become a paradigm for text, vision, and audio tasks (Devlin et al., 2019; Bao et al., 2021; Baevski et al., 2020). In addition to the similarity in the pipeline paradigm, these pre-training models as well have close model structures: On one hand, most of them consist of the following components, *embedding*, *encoder*, *target embedding*, *decoder*, and *target*, on the other hand, many modules in above components are shared among models of different modalities. For example, the transformer module (in encoder component) (Vaswani et al., 2017), which

is successful in the field of text, is increasingly being applied to the vision and audio modalities. (Dosovitskiy et al., 2020; Gulati et al., 2020). Table 1 lists the commonly used pre-training models and their modules.

The trend towards homogeneity in pre-training models is becoming more apparent, which makes it possible to integrate them into a uniform framework. A representative work in this direction is Huggingface Transformers (Wolf et al., 2020), which exploits a non-modular design mode. For each pre-training model in Huggingface Transformers, several separate classes are created, and the code is not refactored with additional abstractions. Users can develop their pre-training models independently which is useful to collaborative development in the community. However, in this design mode, users need to implement the model from scratch when adding a new pre-training model, requiring considerable code work. In addition, with the increased number of pre-training models, the number of classes and lines of code also increases linearly. Codes with the same function may be written many times, which degrades the readability and maintainability of the project.

In response to shortcomings of non-modular design mode, we introduce TencentPretrain, a modular toolkit specially designed for pre-training models of different modalities. As shown in Figure 1, TencentPretrain has five components, namely *embedding*, *encoder*, *target embedding*, *decoder*, and *target*. Among them, *target embedding* and *decoder* components are optional, since the targets of many pre-training models do not involve sequence decoding (Zhang et al., 2020; Lewis et al., 2020). TencentPretrain is hierarchical modular designed with two degrees of freedom. At component level, users are free to combine modules within a component, for example, combining multiple modules in *target* component to perform multi-task pre-training (Lan et al., 2019; Sun et al., 2020). At

* Corresponding author.
E-mail: nlpzhezhaotencent.com

Pre-training model	Modality	Embedding	Encoder	Target embedding	Decoder	Target
ELMo (Peters et al., 2018)	Text	word	bi-lstm	-	-	bilm
InferSent (Conneau et al., 2017)	Text	word	gru	-	-	cls
CoVe (McCann et al., 2017)	Text	word	lstm	word	lstm	lm
BERT (Devlin et al., 2019)	Text	word, pos, seg	transformer	-	-	mlm, sp
GPT-2 (Radford et al., 2019)	Text	word, pos	transformer	-	-	lm
T5 (Raffel et al., 2020)	Text	word	transformer	word	transformer	lm
ViT (Dosovitskiy et al., 2020)	Vision	patch, pos	transformer	-	-	cls
BEiT (Bao et al., 2021)	Vision	patch, pos	transformer	-	-	mlm
S2T (Wang et al., 2020)	Audio	speech, pos	transformer	word, pos	transformer	lm
ViLT (Kim et al., 2021)	Text-vision	word_patch, pos, seg	transformer	-	-	mlm, cls

Table 1: Typical pre-training models and their modules. The above models use different variants of transformer. Due to the page limit, we do not list the details of transformer module in encoder component. In addition, abbreviations are used in embedding and target columns. pos and seg respectively stand for position and segment embeddings. bilm, cls, lm, mlm, sp respectively stand for bi-directional language model, classification, language model, masked language model, sentence prediction.

the model level, users can combine modules from different components to constitute a complete pre-training model.

Modularity in design makes TencentPretrain scalable with the increasing number of newly proposed pre-training models. Users are allowed to reuse existing modules with little efforts, avoiding repeated implementation of core functions. At the same time, TencentPretrain provides a robust and clear interface among different components. It brings flexibility, allowing users to build custom model structures through a configuration file without any code work.

TencentPretrain is implemented with PyTorch (Paszke et al., 2019), and it supports distributed training and DeepSpeed optimization library (Rasley et al., 2020). TencentPretrain is fully connected with Huggingface Transformers, providing comprehensive conversion scripts of pre-training models between the two frameworks. Users can switch between the two frameworks at low cost. TencentPretrain is tested on text, vision, and audio benchmarks and is able to reproduce the results of SOTA pre-training models. The TencentPretrain toolkit is publicly available at <https://github.com/Tencent/TencentPretrain>.

2 Related Work

2.1 Pre-training models

Pre-training models have been widely applied in text scenario. The success of pre-training is largely due to the powerful encoders for feature extraction (e.g., LSTM and Transformer), as well as the progress of pre-training target for learning knowledge from unsupervised corpus (Zhang et al., 2020; Lewis et al., 2020; Lan et al., 2019). More recently,

the text pre-training paradigm has been replicated in other modalities. For example, Transformer encoder (and its variants) has been widely used in vision (Dosovitskiy et al., 2020), audio (Gulati et al., 2020; Chen et al., 2022), and vision-language tasks (Radford et al., 2021; Kim et al., 2021). Regarding pre-training target component, text models have inspired models of other modalities. Mirroring the idea of masked language modeling (MLM), MAE (He et al., 2022), BEiT (Bao et al., 2021), and SimMIM (Xie et al., 2022) use masked image modeling (MIM) for self-supervised vision pre-training. Speech model Wav2vec2.0 (Baevski et al., 2020) exploit negative sampling in pre-training target, which is previously used in word embedding (Mikolov et al., 2013) and sentence prediction models (Logeswaran and Lee, 2018; Devlin et al., 2019; Lan et al., 2019).

In addition to the sharing of modules, several works have recently shown the feasibility of using the same pre-trained weight to handle different modalities simultaneously. For example, ERNIE-ViLG (Zhang et al., 2021) and Talk2Face (Li et al., 2022) exploit prefix language model to achieve bi-directional text-and-image generation. PolyViT uses a single transformer model for image, video and audio classification (Likhoshesterov et al., 2021).

It can be seen that the trend towards homogeneity of pre-training models is becoming obvious, from sharing modules, to using the same network and parameters. This inspires us to build a unified framework that can implement various pre-training models efficiently.

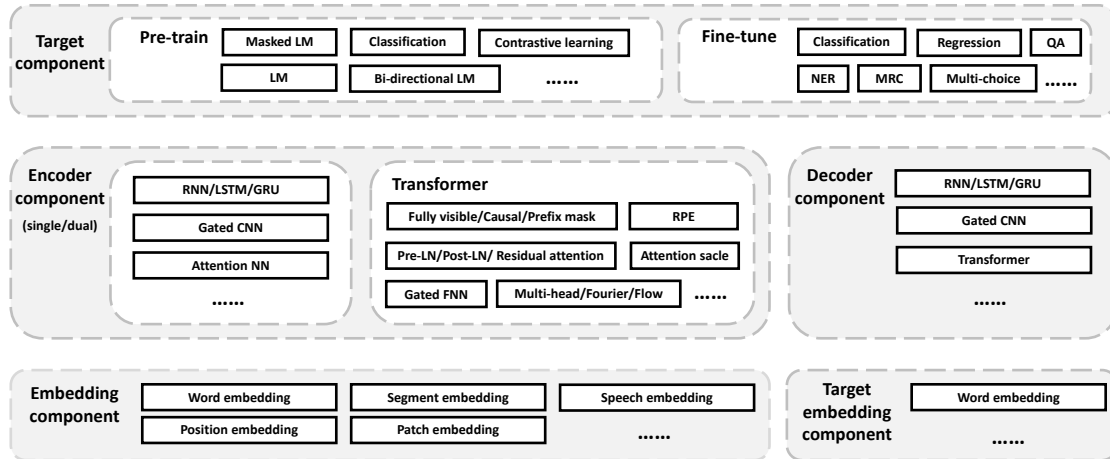


Figure 1: The architecture of TencentPretrain. Pre-training models are implemented through module combination. TencentPretrain encourages reusing the existing modules and writing code at the module granularity.

2.2 Toolkits with modular design

Modular design regards a complex system as the combination of multiple modules, each of which can be independently modified and replaced. In the field of artificial intelligence, a typical work with modular design is Keras (Chollet et al., 2015). The core data structure of Keras is layer. Keras allows building arbitrary graphs of layers to construct NN models. In the NLP field, modular toolkits are prevailing and they decompose models from different perspectives with different abstraction levels. For example, OpenNMT (Klein et al., 2017) is a modular toolkit designed for NMT. It builds an NMT model through the combination of encoder and decoder modules. Related NLP modular toolkits include OpenAttack (designed for text attack) (Zeng et al., 2021), Ngram2vec (designed for word embedding) (Zhao et al., 2017), TextFlint (designed for robustness evaluation) (Wang et al., 2021), NeuralClassifier (designed for text classification) (Liu et al., 2019a), and etc.

Inspired by the above-mentioned works, this paper proposes TencentPretrain, a modular designed toolkit for pre-training models of different modalities. Compared with Huggingface Transformers (Wolf et al., 2020), the most well-known pre-training toolkit, TencentPretrain provides additional abstractions on pre-training model implementations, splitting a complete model into multiple modules hierarchically. Pre-training weights between two toolkits can be switched easily. In fact, TencentPretrain can be regarded as the high-level encapsulation of Huggingface Transformers.

It is worth mentioning that TencentPretrain

reuses part of the code in UER (Zhao et al., 2019), which is published in 2019 and supports several text pre-training models. Compared with UER, TencentPretrain is improved in three aspects: 1) It supports the modular design within components, providing a more scalable manner to build pre-training models; 2) The *target embedding* and *decoder* components are introduced to support sequence generation; 3) In addition to text, TencentPretrain supports vision, audio, and cross-modal pre-training models. Currently, TencentPretrain supports around 30 pre-training models.

3 Framework

The current mainstream pre-training models are basically similar in structure. In the embedding component, the data is mapped into an embedding matrix. And then the matrix is passed through the encoder. Finally the target layer performs pre-training tasks according to the output of the encoder layer. If the pre-training task requires sequence generation, the decoder is inserted between the encoder and the target.

Figure 1 demonstrates the overall framework of TencentPretrain. It divides a pre-training model into five components, and various modules are provided in each component. In practice, a user firstly selects one or multiple modules from each component (modularization within component), and then combine modules from different components to build a pre-training model (modularization cross components). In the rest of this section, we respectively introduce the above five components and modules included in them.

3.1 Embedding

In the *embedding* component, TencentPretrain converts text, image, and audio modal data into embedding matrix. The matrix holds the low-level features as the input to the encoder.

TencentPretrain also contains auxiliary embedding modules, e.g., position embedding and segment embedding. The embedding of pre-training model is usually obtained by the addition of multiple modules. As shown in Table 1 (Embedding column), the addition of word, position, and segment embeddings constitutes the embedding layer of BERT; the addition of patch and position embeddings constitutes the embedding layer of ViT. TencentPretrain supports hierarchical modular design, enabling users to freely combine modules within *embedding* component to construct the desired embedding layer. This design greatly reduces code redundancy since different models often use similar, instead of identical combinations.

3.2 Encoder

TencentPretrain supports traditional encoders (e.g., LSTM and CNN) (Hochreiter and Schmidhuber, 1997; Kim, 2014), as well as transformer and its variants (e.g., different normalization (He et al., 2021), attention (Lee-Thorp et al., 2021), masking strategies (Dong et al., 2019)). Users can construct customized transformer encoder by combining related options.

In addition, TencentPretrain supports dual-stream encoder, with which the users specify two encoder modules separately. Dual-stream encoder is usually used by models related with semantic search, such as text pair model SBERT (Reimers and Gurevych, 2019) and text-image pair model CLIP (Radford et al., 2021).

3.3 Target embedding and decoder (optional)

The pre-training tasks of some models involve sequence generation. These models require modules in *target embedding* component and *decoder* component. The modules in these two components are identical with the modules in *embedding* component and *encoder* component respectively.

3.4 Target

The module in *target* component receives high-level features obtained from encoder (or decoder) and then uses the features to perform pre-training tasks. Specifically, the target estimates gradients by

objectives and updates the network weights. The target is of vital importance to the performance and has been extensively investigated in the pre-training field (Devlin et al., 2019; Lan et al., 2019; Sun et al., 2020). TencentPretrain supports comprehensive target modules, including language model (Radford et al., 2019), classification (Conneau et al., 2017), contrastive learning (Radford et al., 2021), etc.

Sometimes pre-training models use multiple tasks, e.g., predicting word and sentence relationship simultaneously in BERT and ALBERT. And multi-task is especially common in cross-modal scenario (Kim et al., 2021; Lu et al., 2019; Qi et al., 2020) since pre-training models have to deal with supervision signals from different modalities. The model can learn knowledge from different perspectives through multiple tasks. With the characteristic of hierarchical modular design, TencentPretrain facilitates the implementation of multi-task pre-training models. One can introduce multiple tasks by combining different modules in *target* component. The pre-training task can be easily added, modified, and replaced.

3.5 Downstream task fine-tuning

TencentPretrain supports comprehensive downstream tasks, including classification, regression, sequence labeling, reading comprehension, question answering, automated speech recognition, etc. As shown in Figure 1, the downstream task model can be constructed by replacing the pre-training target with specific task. In evaluation section, we show the performances of TencentPretrain on a range of benchmarks.

4 Usage

This section provides examples of building pre-training models with TencentPretrain. The modular design enables the users to quickly build the pre-training model through the combination of modules. Modules used in pre-training models are specified in configuration files and the examples are shown as follows¹:

```
# BERT implementation
{
  "embedding": ["word", "pos", "seg"],
  "encoder": "transformer",
  "target": ["mlm", "sp"]
}
```

¹Due to the page limit, we do not list entire configuration files. More details (e.g., Transformer encoder options) can be found in TencentPretrain project.


```

# T5 implementation
{
  "embedding": ["word"]
  "encoder": "transformer"
  "tgt_embedding": ["word"]
  "decoder": "transformer"
  "target": ["lm"]
}

# ViLT implementation
{
  "embedding": ["patch_word", "pos", "seg"]
  "encoder": "transformer"
  "pooling": "first"
  "target": ["cls", "mlm"]
}

# CLIP implementation
{
  "stream_0":{
    "embedding": ["word", "pos"],
    "encoder": "transformer",
    "pooling": "first"
  }
  "stream_1":{
    "embedding": ["patch", "pos"],
    "encoder": "transformer"
    "pooling": "first"
  }
}
"target": ["clr"]
}

```

- BERT configuration file provides modules in *embedding*, *encoder*, and *target* components. Since BERT has two pre-training tasks, its target is the combination of masked language model (mlm) and sentence prediction (sp).
- T5 involves text generation. Its configuration file specifies modules used in *target embedding* and *decoder* components.
- ViLT, an image-text pre-training model, is basically similar with text pre-training model BERT. The main difference is that an image-text embedding module is used in *embedding* component.
- CLIP is a dual-stream model. The modules in stream0 process text and the modules in stream1 process image. Contrastive learning (clr) module is used in *target* component.

If the desired pre-training model cannot be built by the combination of existing modules, TencentPretrain encourages users to develop a new module, and combine it with existing modules. We take the implementation of ASR model S2T (Wang et al., 2020) as an example. Most modules required by S2T are available and we only need to implement

a new module, speech embedding, which greatly speeds up the implementation process.

TencentPretrain and Huggingface Transformers are interoperable. The conversion scripts are publicly available², and the weights of different pre-training models can be converted between the two frameworks. In practical use, users are free to switch between these two frameworks.

With TencentPretrain, we build a pre-trained weight model zoo. Each pre-trained weight has two versions which can be loaded by either TencentPretrain or Huggingface Transformers. Currently, the TencentPretrain model zoo includes over 50 pre-trained weights. We provide pre-training data as well as training details, allowing users to reproduce results with less effort. The weights (pre-trained by TencentPretrain) are currently downloaded over 500 thousand times per month³.

Model	HF	UER	TP
Transformer	1135	749	795
+BERT	+822	+130 (+word_pos_seg, bert)	+149 (+pos_seg, mlm,sp)
+RoBERTa	+696	+92 (+word_pos,mlm)	+0
+GPT-2	+688	+0	+0
+T5	+1008	+17(+word)	+0
+ViT	+493	-	+59(+patch)
+S2T	+824	-	+51(+speech)
+ViLT	+618	-	+15 (+word_patch)

Table 2: The number of code lines required for implementing a new pre-training model. The comment line in code is not counted. For UER and TencentPretrain, the added modules are also listed. Green and violet are used to denote embedding and target modules. Since UER does not support modularization within component, it has to introduce more modules (classes), e.g., word_pos_seg embedding and bert target, which are decomposed into multiple modules in TencentPretrain.

5 Evaluation

This section evaluates TencentPretrain framework quantitatively. Firstly, we compare TencentPretrain with non-modular framework in terms of implementation cost. Then we show that TencentPretrain can reproduce the results of SOTA models on a range of benchmarks.

²<https://github.com/Tencent/TencentPretrain/tree/main/scripts>

³<https://huggingface.co/uer>

For Huggingface account, we inherit UER account instead of using TencentPretrain account.

Model	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	AVG
BERT-base (Ori.) (Devlin et al., 2019)	83.9	90.7	90.7	65.7	92.3	88.9	56.5	88.6	82.2
BERT-base (DistilBERT) (Sanh et al., 2019)	86.7	91.8	89.6	69.3	92.7	88.6	56.3	89.0	83.0
BERT-base (DynaBERT) (Hou et al., 2020)	84.8	92.0	90.9	71.1	92.9	87.7	58.1	89.8	83.4
BERT-base (Metadistil) (Zhou et al., 2022)	84.6	91.2	91.4	71.4	93.0	87.6	58.9	90.2	83.5
BERT-base (Ours)	83.4	91.1	91.2	67.9	92.4	86.5	59.6	89.1	82.6
RoBERTa-large (Ori.) (Liu et al., 2019b)	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	88.9
RoBERTa-large (Ours)	90.4	94.7	92.1	86.6	96.4	90.2	67.0	92.5	88.7

Table 3: The comparison of TencentPretrain with other implementations on GLUE benchmark. We pre-train from scratch and then fine-tune on a range of datasets

5.1 Implementation cost

The number of code lines is used to estimate the implementation cost. We only count the code lines in classes inheriting *nn.Module*. We compare three frameworks, Huggingface Transformers (HF), UER, and TencentPretrain (TP). Huggingface Transformers exploits non-modular design. UER exploits semi-modular design, which doesn't support modularization within component.

When we continue to add new pre-training models (as shown in Table 2 from top to bottom), the number of code lines required by the TencentPretrain is less than the other two toolkits. Take RoBERTa as an example, TencentPretrain does not require any code work since it reuses modules for BERT. UER needs to add `word_pos` module in *embedding* component and `mlm` module in *target* component. Huggingface Transformers builds a series of classes specific to RoBERTa, such as `RoBERTaModel`, `RoBERTaEmbeddings`, `RoBERTaEncoder`, `RoBERTaPooler`, which greatly increases the number of code lines. For other pre-training models, the conclusions are similar. The homogeneity among pre-training models makes the modular design much more advantageous.

In general, the code styles of Huggingface and TencentPretrain are inconsistent. Huggingface creates separate classes for each pre-training model, while TencentPretrain establishes generic modules that are independent of the specific model. Therefore, for most pre-training models, no additional code implementation is required in TencentPretrain.

5.2 Reproducibility

In this section, we follow the experimental settings of original papers. The scripts for running models on benchmarks are organized here⁴, and users can easily reproduce the results in Table 3 and 4.

⁴<https://github.com/Tencent/TencentPretrain/wiki/Competition-solutions>

For text modality, we use GLUE benchmark to test TencentPretrain's performance. BERT-base and RoBERTa-large are used as test models. The results of BERT-base are listed in the first five rows in Table 3. As shown in AVG column, our result is 82.6, which falls into the range of 82.2-83.5 (the lowest and highest results reported by other papers). The average scores reported by DynaBERT and Metadistil are slightly higher than our result. One of the reasons is that development set of RTE only includes 277 instances, which leads to large fluctuations. The RTE results reported by DynaBERT and Metadistil are 3 point higher than our implementation. For RoBERTa-large, we can observe that our implementation results are close to the results reported by original RoBERTa paper.

Table 4 provides the results on vision and audio tasks. ViT (Dosovitskiy et al., 2020) and BEiT (Bao et al., 2021) are used as test models for vision datasets. Top1 accuracy on vision datasets is reported. The original paper of BEiT only reports results on ImageNet. For audio dataset, we report the Automatic Speech Recognition (ASR) results on LibriSpeech with S2T (Wang et al., 2020). Word Error Rate (WER) is shown in Table 4 (bottom). We can observe that the results of TencentPretrain are close to the results reported by original papers.

Model	CIFAR10	CIFAR100	ImageNet 1000
ViT-base	98.95	91.67	83.97
ViT-base(Ours)	98.73	92.12	83.97
BEiT-large	-	-	87.30
BEiT-large(Ours)	-	-	87.24

Model	devclean	devothor	testclean	testothor
S2T	3.8	8.9	4.4	9.0
S2T(Ours)	3.8	9.2	4.1	9.0

Table 4: The comparison of TencentPretrain with original implementations on datasets of vision and audio modalities.

6 Conclusion

This paper presents TencentPretrain, a pre-training toolkit characterized by modular design and multi-modal support. In TencentPretrain, pre-training models of different modalities are regarded as the combination of multiple modules, which is easy to configure and extensible. Furthermore, we quantitatively demonstrate that TencentPretrain facilitates the users to reuse existing modules and decreases the cost of model development. At last, we test TencentPretrain on a range of datasets and show that it can reproduce the SOTA results.

7 Limitations

Although the TencentPretrain pre-training framework has integrated optimization libraries like Deepspeed and Apex, it still lacks support for other components such as Megatron. In the future, we will provide more parallelism modes to achieve efficient training of large-scale language models (LLM).

Acknowledgements

The work was supported by the National Natural Science Foundation of China under Grant No. 62072458. We are grateful to Lusheng Zhang, Xiyayu Li, Jian Kang, Jinwen Luo, Weidong Guo, Jiachi Liu, Jianwei Cui, Dongxiao Huang, Xingyu Bai, Yang Xu, Huanqin Wu, Tongwen Huang, Peng Meng, Yanming Xu, Chunquan Chen, Xuefeng Yang, Qi Ju for code contribution. We also received helpful ideas and feedback from members of the TencentNLP Oteam and Institute of Computer Vision, Shenzhen University.

References

- 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*, 33:12449–12460.
- Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2021. Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations*.
- Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*.
- François Chollet et al. 2015. keras.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.
- 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.
- 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. *Advances in Neural Information Processing Systems*, 32.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- 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. *arXiv preprint arXiv:2005.08100*.
- Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.
- Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. Realformer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 929–943.
- 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. *Advances in Neural Information Processing Systems*, 33:9782–9793.
- Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR.

- Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.
- 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*.
- James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.
- 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.
- Yudong Li, Xianxu Hou, Zhe Zhao, Linlin Shen, Xuefeng Yang, and Kimmo Yan. 2022. Talk2face: A unified sequence-based framework for diverse face generation and analysis tasks. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4594–4604.
- Valerii Likhoshesterov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. 2021. Polyvit: Co-training vision transformers on images, videos and audio. *arXiv preprint arXiv:2111.12993*.
- Liqun Liu, Funan Mu, Pengyu Li, Xin Mu, Jing Tang, Xingsheng Ai, Ran Fu, Lifeng Wang, and Xing Zhou. 2019a. Neuralclassifier: an open-source neural hierarchical multi-label text classification toolkit. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–92.
- 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*.
- Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.
- Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Di Qi, Lin Su, Jia Song, Edward Cui, Taroon Bharti, and Arun Sacheti. 2020. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966*.
- 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, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 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, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.
- 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.
- 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*.

- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.
- 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.
- Changan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq s2t: Fast speech-to-text modeling with fairseq. 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: System Demonstrations*, pages 33–39.
- Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 347–355.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, 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.
- Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663.
- Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371.
- Han Zhang, Weichong Yin, Yewei Fang, Lanxin Li, Boqiang Duan, Zhihua Wu, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-vilg: Unified generative pre-training for bidirectional vision-language generation. *arXiv preprint arXiv:2112.15283*.
- 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.
- Zhe Zhao, Hui Chen, Jinbin Zhang, Wayne Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. Uer: An open-source toolkit for pre-training models. 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): System Demonstrations*, pages 241–246.
- Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 244–253.
- Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. Bert learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7037–7049.

NeuroX Library for Neuron Analysis of Deep NLP Models

Fahim Dalvi
Qatar Computing Research
Institute, HBKU
faimaduddin@hbku.edu.qa

Hassan Sajjad*
Faculty of Computer Science
Dalhousie University
hsajjad@dal.ca

Nadir Durrani
Qatar Computing Research
Institute, HBKU
ndurrani@hbku.edu.qa

Abstract

Neuron analysis provides insights into how knowledge is structured in representations and discovers the role of neurons in the network. In addition to developing an understanding of our models, neuron analysis enables various applications such as debiasing, domain adaptation and architectural search. We present *NeuroX*, a comprehensive open-source toolkit to conduct neuron analysis of natural language processing models. It implements various interpretation methods under a unified API, and provides a framework for data processing and evaluation, thus making it easier for researchers and practitioners to perform neuron analysis. The Python toolkit is available at <https://www.github.com/fdalvi/NeuroX>.¹

1 Introduction

Interpretation of deep learning models is an essential attribute of trustworthy AI. Researchers have proposed a diverse set of methods to interpret models and answered questions such as: what linguistic phenomena are learned within representations, and what are the salient neurons in the network. For instance, a large body of work analyzed the concepts learned within representations of pre-trained models (Belinkov et al., 2017; Liu et al., 2019; Tenney et al., 2019; Rogers et al., 2020) and showed the presence of core-linguistic knowledge in various parts of the network. Several researchers have carried out this interpretation at a fine-grained level of neurons e.g. Durrani et al. (2020); Torroba Henigen et al. (2020); Antverg and Belinkov (2021) highlighted salient neurons w.r.t any linguistic property in the model and Lundberg and Lee (2017); Dhamdhere et al. (2018); Janizek et al. (2020) identified a set of neurons responsible for a given prediction. At a broader level, these methods can be

*The work was done while the author was at QCRI.

¹Demo Video available here: <https://youtu.be/mLhs2YMx4u8>

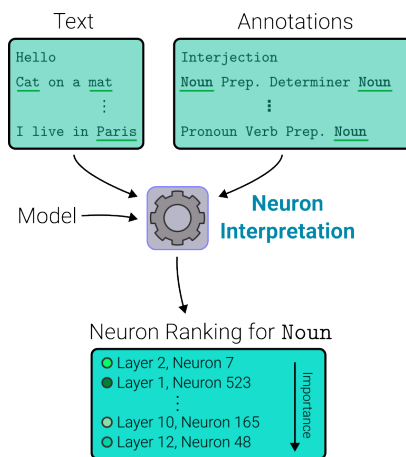


Figure 1: Simplified overview of Neuron Interpretation. Given an annotated text corpus, neuron interpretation methods aim to provide a ranking of neurons in a model w.r.t to their importance to one or more annotated properties (for e.g. "Noun" in this instance)

categorized into representation analysis, neuron analysis and feature attribution methods respectively. Sajjad et al. (2022) provides a comprehensive survey of these methods.

A number of toolkits have been proposed to facilitate the interpretation of deep learning models. For instance, diagNNose (Jumelet, 2020) provides representation analysis and attribution methods. LIT (Tenney et al., 2020) can be used to visualize attention and counterfactual explanations using feature attribution methods. Captum (Kohlikeyan et al., 2020) integrates a large set of attribution methods under a consistent API. All these tools facilitate the interpretability of models. However, due to the diverse ways of interpreting models, they do not cover all sets of methods. Specifically, most of the toolkits do not cover *neuron interpretation* methods that discover and rank neurons with respect to a concept.

Neuron interpretation analyzes and gives insight into how knowledge is structured within a representation. It discovers neurons with respect to a

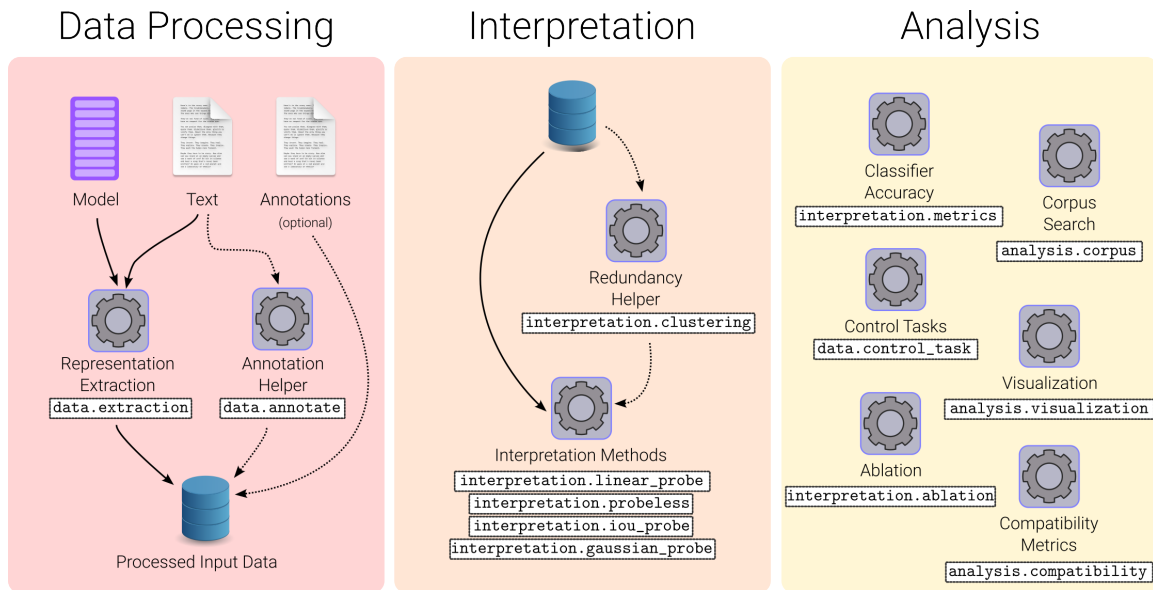


Figure 2: Overall design and architecture of the NeuroX toolkit, with references to their corresponding Python modules in the white boxes.

concept and provides a fine-grained interpretation of deep models. Figure 1 provides a simplified high-level overview of how neuron interpretation methods operate. Given a model, some text and annotations, these methods output a ranking of neurons with respect to their importance to one or more annotated concepts. The ability to interpret neurons enables applications such as debiasing models, controlling predictions of the models on the fly (Bau et al., 2019; Suau et al., 2020), neural architectural search (Dalvi et al., 2020) and domain adaptation (Gu et al., 2021). To make neuron interpretation more accessible, we propose *NeuroX*, an open-source Python toolkit to facilitate neuron interpretation of deep natural language processing (NLP) models.

NeuroX consists of three major components: i) data processing, ii) interpretation and iii) analysis. The data processing implements various ways to generate and upload data for analysis, extract activations and save them efficiently. The interpretation module implements six interpretation methods belonging to two different classes of methods. The analysis module brings together qualitative and quantitative methods to evaluate and visualize the discovered neurons. Figure 2 shows these components and how they interact with each other. We describe them in detail in the following sections. The toolkit itself is compatible with HuggingFace’s transformers (Wolf et al., 2020) API and supports all transformer-based models.

To the best of our knowledge, *NeuroX* is the first toolkit that enables the interpretation of deep NLP models at the neuron level. It serves as a backbone to rapidly test new interpretation techniques using a unified framework and enables comparison and consistent evaluation of these techniques. The toolkit is easy to install and run:

```
pip install neurox
```

with detailed documentation is available at <https://neurox.qcri.org/docs>, including tutorials that showcase various capabilities of the toolkit to quickly get started with neuron interpretation.

2 Data Processing

The data module is responsible for preparing all the inputs for the Interpretation and Analysis modules, as well as filtering the datasets to probe and interpret specific phenomena. As shown in Figure 2, the required inputs to the toolkit are: i) a model and ii) a text corpus annotated towards the property of interest (e.g. data annotated towards toxic word spans in the hate-speech-detection task). The interpretation module can then extract a neuron ranking, highlighting the saliency of the neurons in the model that capture this phenomenon. If annotations are not available, an annotation helper module is made available in the toolkit that can annotate tokens based on arbitrary phenomena e.g. suffixation, lexical properties, or using pre-existing vocabularies. Below we describe the various components of the data module in detail.

Tokenizer	Input Sentence	Tokenized Sentence
bert-base-cased gpt2	"A good-looking house"	"[CLS] A good - looking house [SEP]"
bert-base-cased gpt2	"A good-looking house"	"A Ä good - looking Ä house"
bert-base-cased gpt2	"Mauritians"	"[CLS] ma ##uri ##tian ##s [SEP]"
bert-base-cased gpt2	"Mauritians"	"M aur it ians"
flaubert/flaubert_base_cased	"sport qu' on"	"<s> sport</w> qu</w> '</w> on</w> </s>"
flaubert/flaubert_base_cased	"sport qu'"	"<s> sport</w> qu'</w> </s>"

Table 1: Tokenizers from different models tokenize the same input very differently, sometimes adding special characters at the first subword, or prefixing all subwords except the first subword etc. Sometimes the same model tokenizes the same word (qu') differently depending on the context.

2.1 Representation Extraction

Central to any neuron interpretation method are the neuron activations themselves, i.e. the magnitude of a neuron for any given input. While modern frameworks such as PyTorch and Tensorflow facilitate the extraction of intermediate neuron values for specific models via hooks, it is non-trivial to enable this generically, as the code to extract activations from specific network components (e.g. layers) is highly dependent on the underlying model implementation. *NeuroX* implements generic extractors for specific popular frameworks and provides a highly-configurable PyTorch-based extractor.

Framework Specific Extractors An example of a framework specific extractor is one for Hugging-Face's *transformers* models. The *transformers* library exposes the intermediate output at each layer, which can then be used to access each neuron's (layer output) activation for any given input.

Generic Extractors Apart from framework specific extractors, the toolkit offers a generic extractor for PyTorch model, which runs as a two step process. In the first step, the toolkit maps out the architecture of the given model, and provides the user a json file that contains all the components of the model. The user can then choose exactly which of the components they need the activations for, which are then saved in the second step.

Segmentation and De-Segmentation A unique problem to text and NLP models is that of tokenization. For instance, every *transformers* model has an associated *tokenizer*, that breaks the tokens in an input sentence into subwords depending on the model's vocabulary. The same input can be tokenized differently by each model. To get a neuron's activation for a given input token regardless of tokenization, *NeuroX* runs a detokenization procedure to combine the activation values on subwords into a single activation value. Table 1 shows examples

of how a sentence (and sometimes a word) can be tokenized differently depending on the underlying tokenizer and context. The toolkit also offers the user a choice on how the activation values across subwords should be combined such as `first` or `last` subword or `average` across subwords.

2.2 Annotation Helper

While annotations are available for some linguistic properties, labeled data sets may not always be available. To carry out an interpretation in such a scenario, *NeuroX* offers a helper module that can label the data with a positive or negative label per token. `data.annotate.annotate_data` can annotate each token positively in three different ways:

1. **Preset Vocabulary:** The token exists in the given vocabulary.
2. **Regular expression:** The token matches with the given regular expression. For example, the expression `^\d+$` annotates all tokens that are composed of digits as positive samples.
3. **Python function:** A function that returns a binary True/False for a given token. Arbitrary computation can be done inside this function. For instance, `lambda token: token.endswith("ing")` annotates all tokens ending with *-ing* positively.

3 Interpretation Module

The central module in the *NeuroX* toolkit is the interpretation module, which provides implementations of several neuron and representation analysis methods. Table 2 shows a list of methods that are currently implemented in the toolkit, along with details of what each method's implementation supports.

The method implementations follow a consistent API to make it easy for the user to switch

Interpretation Method	Description	Supports Representation Analysis	Requires Training	Supports multi-class analysis
Linear Probe	Class of probing methods that use a linear classifier for neuron analysis. Specifically, the implementation provides probes introduced by <ul style="list-style-type: none"> • Radford et al. (2019) (Classifier with L1 regularization) • Lakretz et al. (2019) (Classifier with L2 regularization) • Dalvi et al. (2019a) (Classifier with Elastic Net regularization) 	Yes	Yes	Yes
Probeless	A corpus-based neuron search method that obtains neuron rankings based on an accumulative strategy, introduced by Antverg and Belinkov (2021)	No	No	Yes
IoU Probe	A mask-based method introduced by Mu and Andreas (2020) that computes Intersection over Union between tokens representing a specific concept and tokens that have high activation values for specific neurons	No	No	No
Gaussian Probe	A multivariate Gaussian based classifier introduced by Torroba Hennigen et al. (2020) that can probe for neurons whose activation values follow a gaussian distribution.	Yes	Yes	Yes
Mean Select	A corpus-based neuron ranking method introduced by Fan et al. (2023) that derives neuron importances by looking at activation values across contexts where a concept appears.	No	No	Yes

Table 2: An overview of the neuron interpretation methods currently implemented in the *NeuroX* toolkit.

between them. Specifically, each method at least implements the following functions:

- `method.train_probe`: This function takes in the pre-processed data (extracted activations, prepared labels etc) as described in section 2, and returns back a probe that can be used to perform neuron analysis. Some methods do not require any training, in which case this function just stores the input for future use.
- `method.evaluate_probe`: This function takes an evaluation set and returns the performance of the probe on the given set. The evaluation set itself can be a control task, and the output score can be computed using several pre-implemented metrics. Section 4 discusses the various evaluation metrics in detail.
- `method.get_neuron_ordering`: This function returns an ordering/ranking of all the neurons being analyzed with respect to their importance to the task at hand. For instance, if the probe was trained to analyze *Nouns*, this function will return a sorted list of neurons (by importance) that activate for *Nouns* in the given dataset.

The interpretation methods themselves may be

able to probe multiple properties at the same time (multi-class probing), or only a single concept (binary probing). Additionally, some interpretation methods can also perform representation-level analysis, i.e. probe an entire layer rather than individual neurons.

Redundancy Analysis: Dalvi et al. (2020) have shown that large neural networks learn knowledge redundantly where multiple neurons are optimized to activate on the same input. This is encouraged by the optimization choices such as dropouts which explicitly force neurons to learn in the absence of other neurons. In order to facilitate the analysis of redundant neurons, the toolkit provides a clustering based non-redundant neuron extraction method. Running the neurons through `interpretation.clustering.extract_independent_neurons` first before performing any probing can reduce the overall search space of neurons, and lead to better findings and analyses.

4 Analysis and Evaluation

The analysis module provides implementations of various evaluation and analysis methods. Some of these methods provide quantitative results like

accuracy scores, while others allow users to perform qualitative analysis on neurons.

4.1 Classifier Accuracy

Classifier accuracy reciprocates the probing framework (Belinkov et al., 2017; Hupkes et al., 2018). Once a neuron ranking is obtained, a classifier is trained towards the task of interest (the intrinsic concept for which the probe was originally trained) with the selected neurons. The delta between oracle performance (accuracy using all the neurons) and the accuracy of the classifier using the selected neurons measures the efficacy of the ranking.

Selectivity It is important to ensure that the probe is truly representing the concepts encoded within the learned representations and not memorizing them during classifier training. *NeuroX* enables control task selectivity, a measure proposed by Hewitt and Liang (2019) to mitigate memorization using the `data.control_task` module.

4.2 Ablation

An alternative approach used by (Dalvi et al., 2019a) is to ablate all but the selected neurons in the trained probe. The `interpretation.ablation` allows manipulating the input data by keeping/filtering specific neurons in the order of their importance, allowing users to measure the drop in performance with selected neurons.

4.3 Mutual Information

Information theoretic metrics such as mutual information have also been used to interpret representations of deep NLP models (Pimentel et al., 2020). Here, the goal is to measure the amount of information a representation provides about a linguistic concept. It is computed by calculating the mutual information between a subset of neurons and linguistic concepts.

4.4 Compatibility Metrics

Another set of evaluation metrics recently proposed by Fan et al. (2023) carries out a pair-wise comparison of the discovered neurons across methods. While this strategy does not provide a direct evaluation of a neuron interpretation method, it provides an insight into how *compatible* a method is with the other available methods. *NeuroX* implements two compatibility metrics in the `analysis.compatibility` module: i) Average Overlap (which shows how aligned a method is

with others) and ii) NeuronVote (which shows how well-endorsed the ranking of a method is by other methods).

4.5 Qualitative Evaluation

Visualizations have been used effectively to gain qualitative insights on analyzing neural networks (Karpathy et al., 2015; Kádár et al., 2017). *NeuroX* provides a text visualization module (`analysis.visualization`) that displays the activations of neurons w.r.t. to a concept (e.g. Figure 3). The toolkit also allows corpus-based analysis in the `analysis.corpus` module by extracting the top n words in a corpus that activate a neuron. Examples are shown in Table 3.

Mr. Gotlieb , who **erves** as a consultant to Stikeman , Elliott , one of Canada 's **biggest** law firms

From a helicopter a thousand feet above Oakland after the **deadliest** earthquake in U.S. history

Traders said property-casualty companies with the **heaviest** exposure in the San **Francisco** area include the OTC 's Safeco and Ohio Casualty .

(a) Superlative Adjective Neuron

It has been **trying** to improve its share of the residential market .

He **declined** to comment on whether the company is **considering** a dividend or is **planning** any acquisition .

The university is **considering** installing a \$ **250,000** system to store applications electronically .

(b) Gerund Verb Neuron

Figure 3: Visualizations (POS) – Superlative Adjective and Gerund Verb Neurons

5 Miscellaneous Functions

5.1 Scalability

Extracting, saving and working with neuron activations over large datasets and models can be very expensive, since each neuron’s activation is saved for each token in the input corpus. To enable both disk- and runtime-savings, *NeuroX* provides a low precision mode where all the activations are saved using 16-bit precision instead of the default 32/64-bit precision. This results in considerable storage/memory savings and also improves training/inference performance depending on the method and underlying hardware. The precision can be controlled by supplying the `dtype=float16` option to the extraction/interpretation methods.

5.2 Disk Formats for Representations

The toolkit offers flexibility to the user over the format used to save the neuron activations. Specifi-

Neuron	concept	Model	Top-5 words
Layer 9: 624	VBD	RoBERTa	supplied, deposited, supervised, paled, summoned
Layer 2: 750	VBG	RoBERTa	exciting, turning, seeing, owning, bonuses
Layer 0: 249	VBG	BERT	requiring, eliminating, creates, citing, happening
Layer 1: 585	VBZ	XLNet	achieves, drops, installments, steps, lapses, refunds
Layer 2: 254	CD	RoBERTa	23, 28, 7.567, 56, 43
Layer 5: 618	CD	BERT	360, 370, 712, 14.24, 550
Layer 1: 557	LOC	XLNet	Minneapolis, Polonnaruwa, Mwangura, Anuradhapura, Kobe
Layer 5: 343	ORG	RoBERTa	DIA, Horobets, Al-Anbar, IBRD, GSPC
Layer 10: 61	PER	RoBERTa	Grassley, Cornwall, Dalai, Bernanke, Mr.Yushchenko
Layer 6: 132	PER	BERT	Nick, Manie, Troy, Sam, Leith
Layer 2: 343	YOC	BERT	1897, 1918, 1901, 1920, Alam

Table 3: Ranked list of words for some individual neurons, VBD: Past-tense verb, VBG: Gerund Verb, VBZ: Third person singular, CD: Numbers, LOC: Location, ORG: Organization, PER: Person, YOC: Year of the century

cally, it offers readers and writers for a text-based format (json) and a binary format (hdf5). The binary format provides faster saving/loading performance, speeding up experiments with a large number of neurons or a large amount of text. On the other hand, the text-based format is considerably easier to debug.

6 Related Work

A number of toolkits have been made available to carry out the analysis and interpretation of neural network models. The What-If tool (Wexler et al., 2019) inspects machine learning models and provides users an insight into the trained model based on the predictions. Seq2Seq-Vis (Strobelt et al., 2018) enables the user to trace back the prediction decisions to the input in neural machine translation models. Captum (Kokhlikyan et al., 2020) provides generic implementations of a number of gradient and perturbation-based attribution algorithms. LIT (Tenney et al., 2020) implements various methods of counterfactual explanations, attribution methods and visualization of attention. diagNNose (Jumelet, 2020) integrates representation analysis methods and attribution methods and finally, iModelsX (Singh and Gao, 2023) aims to provide natural explanations for datasets, which can provide insights into the models that are trained with these datasets. While these tools cover a number of interpretation methods, none of them facilitate neuron-level interpretation of NLP models. The LM-Debugger toolkit (Geva et al., 2022) is an interactive debugger for transformer LMs, which provides a fine-grained interpretation and a powerful framework for intervening in LM behavior.

Ecco (Alammar, 2021) is a visualization based library that implements saliency methods and ad-

ditionally enables visualization of neurons of the network. Similar to Ecco, the *NeuroX* toolkit enables visualization of neurons of the network. In addition, we implement a wide range of neuron interpretation methods that can be accessed using a uniform API and provide various analysis and evaluation methods. Our toolkit empowers researchers to focus on specific parts of the neuron interpretation research such as interpretation, comparison or evaluation without worrying about setting up the rest of the pipeline like data processing, embedding extraction, integration with various pre-trained models, and evaluation of the method. *NeuroX* powers other interpretation analysis frameworks such as ConceptX (Alam et al., 2022) and NxPlain (Dalvi et al., 2023).

The previous version of *NeuroX* (Dalvi et al., 2019b) only supported a specific machine translation library and one neuron interpretation method (Dalvi et al., 2019a) as a GUI app. The current Python toolkit is a redesigned version with a unified architecture. It includes multiple features like a data processing module, numerous neuron interpretation and evaluation methods, and seamless integration with popular toolkits such as HuggingFace’s *transformers*.

7 Conclusion and Future Work

We presented *NeuroX*, an open-source toolkit to carry out neuron-level interpretation of representations learned in deep NLP models. It maintains implementations of several neuron analysis methods under a consistent API, and provides implementations for preparing the data, analyzing neurons and evaluating the methods. In the future, *NeuroX* plans to expand its extraction module to other frameworks like FairSeq and OpenNMT-py. In ad-

dition, we plan to integrate attribution based neuron saliency methods to add another class of interpretation methods to the toolkit.

8 Acknowledgements

We are grateful to all NeuroX contributors and users who have provided bug reports to improve the toolkit. Specifically, we would like to thank Ahmed Abdelali for testing and reporting bugs with the extraction implementations, David Arps for scalability improvements, Yimin Fan for implementing several interpretation methods and Yifan Zhang for working on detailed documentation and tutorials.

9 Ethical Considerations

The NeuroX toolkit provides a post hoc interpretation of pre-trained models. The toolkit makes a contribution towards improving the transparency of deep models and may discover biases present in these models. We do not foresee any direct ethical issues with respect to the developed toolkit. In terms of the neuron interpretation methods, the majority of them are based on the correlation between neurons and the input. One potential issue with such an interpretation is its faithfulness with respect to the knowledge used by the model in making predictions. However, this is not a limitation of the toolkit but a limitation of the research methods in general.

References

- Firoj Alam, Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Abdul Rafae Khan, and Jia Xu. 2022. [Conceptx: A framework for latent concept analysis](#).
- J Alammar. 2021. [Ecco: An open source library for the explainability of transformer language models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online. Association for Computational Linguistics.
- Omer Antverg and Yonatan Belinkov. 2021. On the pitfalls of analyzing individual neurons in language models. In *International Conference on Learning Representations*.
- Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. [Identifying and controlling important neurons in neural machine translation](#). In *International Conference on Learning Representations*.
- Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. [What do Neural Machine Translation Models Learn about Morphology?](#) In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver. Association for Computational Linguistics.
- Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D. Anthony Bau, and James Glass. 2019a. [What is one grain of sand in the desert? analyzing individual neurons in deep nlp models](#). In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI, Oral presentation)*.
- Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Tamim Jaban, Mus’ab Husaini, and Ummar Abbas. 2023. [NxPlain: A web-based tool for discovery of latent concepts](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 75–83, Dubrovnik, Croatia. Association for Computational Linguistics.
- Fahim Dalvi, Avery Nortonsmith, D Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, and James Glass. 2019b. [Neurox: A toolkit for analyzing individual neurons in neural networks](#). *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.
- Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. [Analyzing redundancy in pretrained transformer models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP-2020)*, Online.
- Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. 2018. [How important is a neuron?](#) *CoRR*, abs/1805.12233.
- Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. [Analyzing individual neurons in pre-trained language models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.
- Yimin Fan, Fahim Dalvi, Nadir Durrani, and Hassan Sajjad. 2023. [Evaluating neuron interpretation methods of nlp models](#).
- Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022. [LM-debugger: An interactive tool for inspection and intervention in transformer-based language models](#). In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.
- Shuhao Gu, Yang Feng, and Wanying Xie. 2021. [Pruning-then-expanding model for domain adaptation of neural machine translation](#).

- John Hewitt and Percy Liang. 2019. [Designing and interpreting probes with control tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China.
- Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. [Visualisation and ‘diagnostic classifiers’ reveal how recurrent and recursive neural networks process hierarchical structure](#).
- Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. 2020. [Explaining explanations: Axiomatic feature interactions for deep networks](#).
- Jaap Jumelet. 2020. [diagNNose: A library for neural activation analysis](#). In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 342–350, Online. Association for Computational Linguistics.
- Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. [Representation of linguistic form and function in recurrent neural networks](#). *Computational Linguistics*, 43(4):761–780.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. [Visualizing and understanding recurrent networks](#). *arXiv preprint arXiv:1506.02078*.
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. [Captum: A unified and generic model interpretability library for PyTorch](#).
- Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. [The emergence of number and syntax units in LSTM language models](#). 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 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.
- Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. [Linguistic knowledge and transferability of contextual representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.
- Scott M Lundberg and Su-In Lee. 2017. [A unified approach to interpreting model predictions](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.
- Jesse Mu and Jacob Andreas. 2020. [Compositional explanations of neurons](#). *CoRR*, abs/2006.14032.
- Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. [Information-theoretic probing for linguistic structure](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. [Language models are unsupervised multitask learners](#). *OpenAI blog*, 1(8):9.
- Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. [A primer in BERTology: What we know about how BERT works](#). *Transactions of the Association for Computational Linguistics*, 8:842–866.
- Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. 2022. [Neuron-level Interpretation of Deep NLP Models: A Survey](#). *Transactions of the Association for Computational Linguistics*.
- Chandan Singh and Jianfeng Gao. 2023. [Emb-GAM: an interpretable and efficient predictor using pre-trained language models](#).
- Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander Rush. 2018. [Debugging sequence-to-sequence models with Seq2Seq-vis](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 368–370, Brussels, Belgium. Association for Computational Linguistics.
- Xavier Suau, Luca Zappella, and Nicholas Apostoloff. 2020. [Finding experts in transformer models](#). *CoRR*, abs/2005.07647.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. [BERT rediscovers the classical NLP pipeline](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.
- Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. [The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.
- Lucas Torroba Hennigen, Adina Williams, and Ryan Cotterell. 2020. [Intrinsic probing through dimension selection](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 197–216, Online. Association for Computational Linguistics.

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, 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](#). pages 38–45. Association for Computational Linguistics.

SCILIT: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation

Nianlong Gu

Institute of Neuroinformatics,
University of Zurich and
ETH Zurich
nianlong@ini.ethz.ch

Richard H.R. Hahnloser

Institute of Neuroinformatics,
University of Zurich and
ETH Zurich
rich@ini.ethz.ch

Abstract

Scientific writing involves retrieving, summarizing, and citing relevant papers, which can be time-consuming processes. Although in many workflows these processes are serially linked, there are opportunities for natural language processing (NLP) to provide end-to-end assistive tools. We propose SCILIT, a pipeline that automatically recommends relevant papers, extracts highlights, and suggests a reference sentence as a citation of a paper, taking into consideration the user-provided context and keywords. SCILIT efficiently recommends papers from large databases of hundreds of millions of papers using a two-stage prefetching and re-ranking literature search system that flexibly deals with addition and removal of a paper database. We provide a convenient user interface that displays the recommended papers as extractive summaries and that offers abtractively-generated citing sentences which are aligned with the provided context and which mention the chosen keyword(s). Our assistive tool for literature discovery and scientific writing is available at <https://scilit.vercel.app>

1 Introduction

When we compose sentences like “Our experiments show that XXX performs significantly worse than YYY” in a manuscript, we may want to find papers that report similar performance evaluations (Cohan et al., 2019) and discuss these in our manuscript. This process is a non-trivial task requiring in-depth human involvement in finding, summarizing, and citing papers, which raises the question whether it is possible to partly automate this process to reduce users’ cognitive load in searching, retrieving, reading, and rephrasing related findings.

Recent advances in natural language processing (NLP) help answer this question. First, releases of large scientific corpora such as S2ORC (Lo et al., 2020) and General Index (Else, 2021) provide opportunities for building large databases of scientific

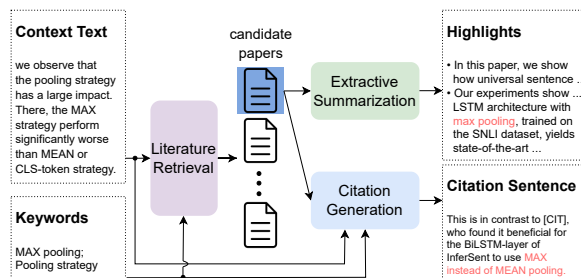


Figure 1: The main workflow of our platform.

papers. Second, such databases can be linked to systems for text retrieval (Guo et al., 2020), citation recommendation (Färber and Jatowt, 2020; Gu et al., 2022b; Medić and Snajder, 2020), extractive summarization (Zhong et al., 2020; Gidiotis and Tsoumakas, 2020; Gu et al., 2022a), and citation generation (Xing et al., 2020a; Ge et al., 2021; Wang et al., 2022), all of which can be tailored to meet the requirements of an author’s manuscript.

To build a comprehensive system that helps authors with finding, reading, and summarizing of literature, the following challenges must be overcome: The system must index many papers (e.g., S2ORC has over 136 million papers (Lo et al., 2020)) to achieve good coverage, it must respond quickly to queries, and it must be flexible to handle database additions and deletions. In addition, the overall architecture should be modular to make it simple to upgrade components when better algorithms become available.

To meet these challenges, we developed SCILIT, a platform for concurrent literature discovery, summarization, and citation generation. We propose a hierarchical architecture for paper retrieval that efficiently retrieves papers from multiple large corpora. On each corpus (e.g., S2ORC and PMCOA (of Medicine, 2003)), we build an efficient prefetching system based on a keyword inverted index and a document embedding index. The prefetched documents are then re-ordered (re-ranked) by a fine-

tuned SciBERT (Beltagy et al., 2019). Such an architecture allows us to dynamically add or remove databases and update one database and its index without significantly affecting the others. From a user-chosen document (i.e., target paper), we extract highlights using a light-weight extractive summarization model proposed in Gu et al. (2022a). Furthermore, using a fine-tuned T5 model (Raffel et al., 2020), we generate a citing sentence based on the abstract of the target paper, the context (the text surrounding the original citation sentence), and the keywords provided by users. We also develop a microservice-based architecture that allows easy updating of algorithms.

In summary, our main contributions are:

- We demonstrate SciLIT, a platform for searching, summarizing, and citing scientific papers.
- We evaluate SciLIT on scientific literature retrieval, paper summarization, and context-aware citation sentence generation, and showcase the generation of a related-work paragraph.
- A live demo website of our system is at <https://scilit.vercel.app> and our implementation and data are at <https://github.com/nianlonggu/SciLit> and a video demonstrating the system can be viewed at <https://youtu.be/PKvNaY5Og1Y>

2 SciLit

Figure 1 shows the workflow of our system. A literature discovery module receives a context text and keywords provided by a user and recommends a list of relevant papers that are semantically similar with the context and that match the keywords used as Boolean filters (Gökçe et al., 2020). For each recommended paper, an extractive summarizer selects a short list of sentences from the full text as highlights. From the target paper selected by the user, a citation generation module takes the abstract together with the context and keywords as inputs and generates a citation sentence that references the target paper and that fits the context and keywords.

We define the context as the text before a citation sentence because we focus on the workflow of first finding papers and then writing citation sentences, rather than finding the missing citation in a

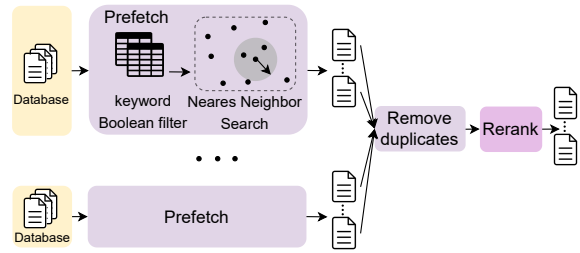


Figure 2: Schematic of literature retrieval. From each database, candidate documents are prefetched by a cascade of keyword boolean filter and embedding-based nearest neighbor search. Then, candidate documents are reranked by a fine-tuned SciBERT.

given sentence as in Gu et al. (2022b); Medić and Snajder (2020). The user-provided keywords are optional. When no keywords are explicitly given during training and evaluation of our system, we use the keywords occurring in both the context, the cited paper, and the citation sentence as a substitutes for user-provided keywords.

2.1 Literature Discovery

The literature discovery module takes as inputs the context and keywords and recommends papers that are worth citing in the current context. To strike a balance between query accuracy and speed on large scientific corpora, our document discovery module employs a two-stage prefetching-ranking strategy (Gu et al., 2022b) (Figure 2). For each scientific corpus, we build a database and create an efficient prefetching model that we use to pre-filter N_p (see the discussion of N_p in Table 2 and Section 3.2) candidate documents based on the provided keywords and context. After removing duplicates, we re-rank the prefetched documents from each database to produce the final order.

Databases. We dump each corpus into a separate SQLite (Hipp, 2000) database to allow flexibility in deploying and updating of independent prefetching servers. We further process documents from different corpora into a unified JSON schema so that we can use the same codebase to index, query, summarize, and display documents from different corpora. The JSON schema includes “Title”, “Author”, etc., for metadata, and “Content.Abstract_Parsed”, “Content.Fullbody_Parsed” for parsed full text, The details are given in Appendix B.

Prefetching. The prefetching model of a given SQLite database consists of an inverted index and an embedding index. The inverted index stores the paper IDs of all publications that contain a given

Corpus	Databases			Inverted Index				Embedding Index	
	# of papers	# papers with fullbody	until date	keywords length	# of keywords	data format	storage size	embedding dimension	storage size
S2ORC	136.60 M	12.44 M	2020-04-14		1.20 B		769 GB		169 GB
PMCOA	2.89 M	2.73 M	2022-06-17	unigram, bigram	0.30 B	sqlitedict	145 GB	256	2.9 GB
arXiv	1.69 M	1.69 M	2022-07-28		0.15 B		77 GB		1.7 GB

Table 1: Statistics of our literature discovery system. We indexed S2ORC (Lo et al., 2020), PMCOA (of Medicine, 2003), and arXiv (Kaggle, 2022), which contain large numbers of recent scientific papers in diverse fields.

keyword, such as a unigram like “computer” or a bigram like “machine learning”, where the paper ID is a unique identifier using which we can retrieve the paper’s content from the database. The embedding index is formed by the embeddings of all papers in the database. Embeddings are 256-dimensional vectors computed by Sent2Vec (Pagliardini et al., 2018) (we simply average the embeddings of all words in a document). We train Sent2Vec using sentences obtained from the full text of the papers contained in S2ORC.

Using the keywords and a specific syntax, we first perform Boolean filtering (Gökçe et al., 2020) of the inverted index. For example, given “POS tag;2010..2022”, we will filter papers published between 2010 and 2022 that mention “POS tag”. The filtered papers are then ranked based on the cosine similarity between the papers’ Sent2Vec embeddings and the context embedding. Such a hybrid of lexical filtering and semantic ranking allows users to find papers that are semantically similar to the context and that flexibly meet a constrained search scope.

Statistics for the database and indexing system are reported in Table 1. Details of the indexing implementation are shown in Appendix C.

Duplicate Removal. Since corpora can overlap, the prefetched candidates from multiple corpora can contain duplicate items. To remove duplicated candidates, we check the title and authors and keep only one record per paper for reranking.

Reranking. We use SciBERT (Beltagy et al., 2019) to rerank prefetched candidates, aiming at highly ranking papers that can be cited given the context and keywords. We follow Gu et al. (2022b) to compute an affinity score as follows: we pass an input text “[CLS]*query*[PAD]*paper*[PAD]” to SciBERT, where the *query* q is a concatenation of the context and the keywords, and *paper* d is a concatenation of the title and the abstract of the candidate paper. The encoded output of the “[CLS]” token is passed to a linear layer, which outputs a scalar $s(q, d)$ that

we interpret as the affinity score between the query q and the paper d . To train the reranker, we use the cross-entropy loss:

$$L = -\log \frac{\exp s(q, d^+)}{\exp s(q, d^+) + \sum_{i=1}^N \exp s(q, d_i^-)}, \quad (1)$$

where d^+ is the paper actually cited in the query, and d_i^- is one of N ($N = 10$) uncited papers that are randomly sampled from prefetched candidate at each training iteration.

2.2 Extractive Summarization

The extractive summarization module extract a short list of sentences from the full text of a paper to highlight the main points to a reader. We choose the summary to be extractive rather than abstractive to prevent readers from being misled by the potential hallucinations introduced in abstractive summarization models (Nan et al., 2021; Xu et al., 2020; Wang et al., 2020). The extractive summarization model must efficiently select sentences from a given document so that users do not experience obvious delays.

In this paper, we employ MemSum, an RNN-based extractive summarizer that models the extraction process as a Markov decision process in a reinforcement learning framework. MemSum has been trained on the PubMed dataset Gu et al. (2022a) and it can summarize long papers without exhausting GPU memory due to its lightweight model structure. Also, MemSum is computationally efficient, taking only 0.1 sec on average to summarize a paper. These features make it a suitable model for our extractive summarization module.

2.3 Citation Generation Module

The citation generation module acts as an abstract summarizer that takes as input the context, the keywords, and the target paper to be cited; it then generates a sentence that cites the target paper and narrates it in context.

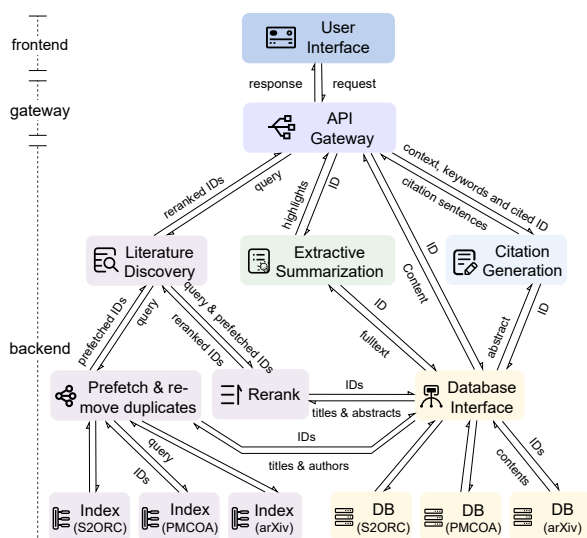


Figure 3: The architecture of our platform. The direction of an arrow represents the direction of data flow.

By providing keywords as inputs to a sequence-to-sequence model, our input differs from previous works on automatic citation generation (Ge et al., 2021; Xing et al., 2020b), which use only the context as inputs but no keywords. We consider keywords to be an important source of input because we believe that authors usually have a clear intention when citing a paper, and a keyword can sometimes more easily convey this intention than a long text. In the case shown in Figure 1, for example, after writing the context “MAX pooling performs worse than MEAN pooling”, the author naturally intends to discuss papers about “MAX pooling”. Therefore, the keyword “MAX pooling” should be used as a thematic cue for citation sentence generation. Moreover, making the citation generation model conditional on keywords also allows users to fine-tune the generated citation text by simply adjusting the keywords, thus making the system interactive and conveniently tunable.

To make the generation conditional on context, keywords, and cited papers, we fine-tuned a T5 (Raffel et al., 2020) so that its input is a concatenation of three attributes: keywords, context, and the abstract of a cited paper, each preceded by a special field name to make attributes distinguishable to the model: `keywords: XXX. context: XXX. target abstract: XXX`. The corresponding decoding output is the actual citation sentence that cites the target paper.

2.4 Microservice-based Architecture

We build our platform as a network of microservices (Figure 3). An API gateway routes requests from the frontend to the target microservice on the backend. The microservices run separate modules on their respective Flask servers (Aggarwal, 2014) and communicate with each other by sending HTTP requests and waiting for responses. When a query request arrives, the API gateway forwards the query to the literature discovery service, which calls the prefetching and reranking services to get the reranked IDs. The API gateway then sends the paper IDs to the extractive summarization service to receive the highlights of each recommended paper. The gateway also sends the context, keywords, and recommended paper IDs to the citation generation service to suggest citation sentences. The database interface service manages the databases of multiple scientific corpora and provides a unified interface to access the paper content given its ID. Each microservice runs in an independent environment, which makes it easy to upgrade backend systems online, such as adding or removing a database or updating an algorithm.

3 Evaluation

In this section, we first show how SCILIT works and then we evaluate its performance.

3.1 Demonstration

Our user interface runs on a web page (Figure 4) created with ReactJS¹. The left sidebar is an input panel where users can enter context and keywords and trigger a query by clicking the *search* button. Retrieved papers are displayed in the search-results panel on the right. Users can scroll up and down or paginate to browse through the recommended papers. Each paper is accompanied by highlights and a suggested citation sentence generated by our extractive summarization and citation generation services, respectively. Users can cite a paper by clicking on the *cite* button and the suggested citation sentence will jump to the editing area on the left where users can tweak the sentence by changing keywords and clicking on the *fine-tune generation* button, or they can edit the sentences manually. Exporting citation information is also supported.

¹<https://reactjs.org/>

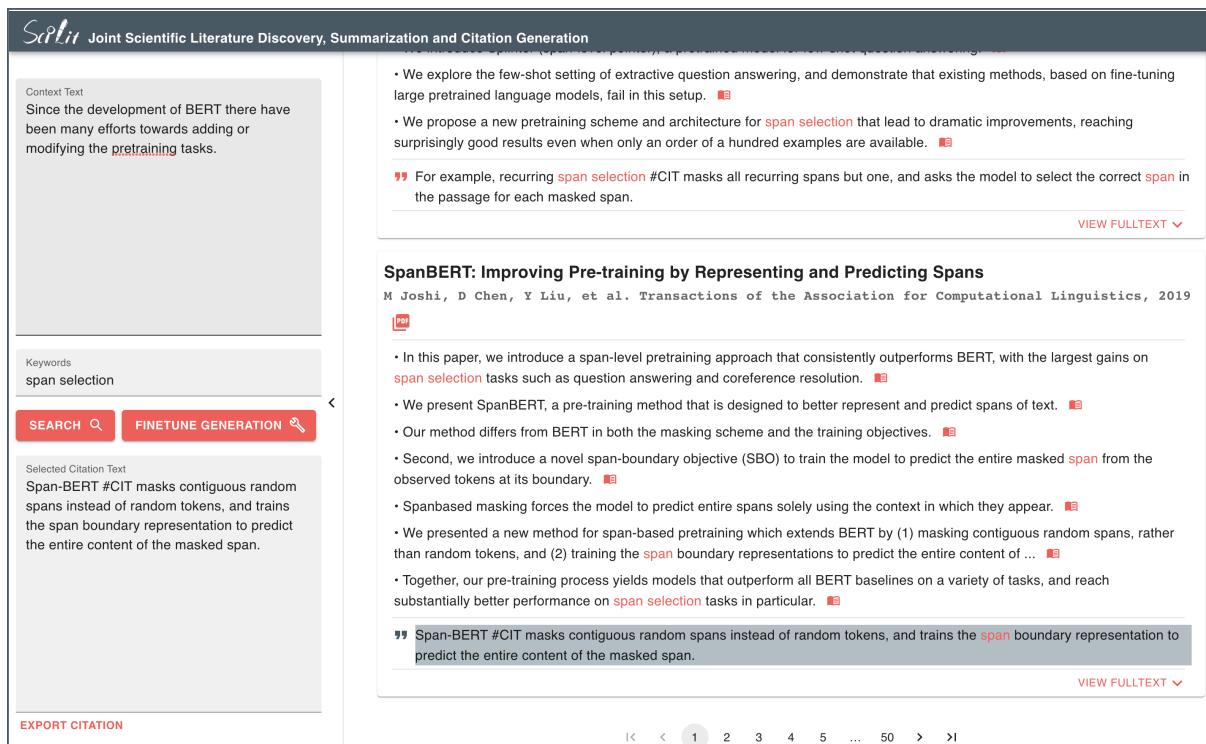


Figure 4: Overview of the user interface. The context text comes from the related work section in Glass et al. (2020).

N_p	time (s/query)	Recall@ K (R@ K)					
		R@1	R@5	R@10	R@20	R@50	R@100
50	2.02	0.107	0.208	0.263	0.305	0.327	0.331
100	2.55	0.096	0.215	0.278	0.328	0.384	0.401
200	3.26	0.095	0.220	0.275	0.339	0.420	0.452
300	3.93	0.095	0.204	0.273	0.330	0.422	0.482

Table 2: Paper retrieval performance measured by the recall of the top K recommendations. N_p denotes the number of prefetched candidates per corpus.

3.2 Performance

Evaluation Dataset. We evaluated SCILIT on a test set containing 1530 samples, mainly from papers published in 2022 in the fields of computer science and biomedical science. Each sample contains the following information: 1) context, up to 6 sentences preceding the citation sentence and within the same section; 2) keywords, up to 2 uni- or bi-grams that occur in all of the context, the citation sentence, and the cited paper; 3) ID of the cited paper; 4) the citation sentence following the context, which is the ground truth for evaluating generated citations. For quality control, we only include citation sentences in the test set that cite one paper.

Paper Retrieval. For each sample in the evaluation dataset, we use context and keywords as queries and invoke the literature search service to

Model	Rouge-1	Rouge-2	Rouge-L
BertSum (Liu, 2019)	42.53	16.89	39.18
MemSum (Gu et al., 2022a)	46.40*	19.61*	42.66*

Table 3: The extractive summarization performance. "*" indicates statistical significance in comparison to baselines with a 95% bootstrap confidence interval.

first prefetch N_p candidates from each of the three corpora (S2ORC, PMCOA, and arXiv). We remove duplicates and then we rank the prefetched candidates. The top K recommendations serve to evaluate the retrieval performance (Table 2). We observed that for large K ($K = 50, 100$), the recall increases as N_p increases, whereas for small K ($K = 5, 10, 20$), the recall first increases and then starts to decrease, indicating that the reranking performance is impacted by more prefetched candidates. We choose $N_p = 100$ as the default value, which was fast and achieved the best performance for R@10.

Extractive Summarization. To evaluate the summaries, following Zhong et al. (2020); Xiao and Carenini (2019), we computed the ROUGE F1 scores between the summary sentences extracted from the full body and the corresponding abstract. MemSum significantly outperformed BertSum (Liu, 2019), a Bert-based summarizer that

generation pipeline	Rouge-1	Rouge-2	Rouge-L
generation-only	32.96	9.19	24.52
Best of top 1 paper	28.62	6.00	21.05
Best of top 5 papers	34.92	9.59	26.23
Best of top 10 papers	36.83*	10.98*	28.10*

Table 4: The performance of citation generation.

retrieval ($N_p = 100$)	R@1	R@5	R@10	R@20	R@50	R@100
w keywords	0.096	0.215	0.278	0.328	0.384	0.401
w/o keywords	0.013	0.050	0.085	0.125	0.199	0.250

citation generation	Rouge-1	Rouge-2	Rouge-L
w keywords	32.96	9.19	24.52
w/o keywords	26.57	5.56	20.39

Table 5: Ablation study on retrieval and citation generation performance.

requires truncation of long documents, indicating the effectiveness of MemSum in extractively summarizing scientific documents.

Citation Generation. To evaluate our joint retrieval and citation generation pipeline, we let our system first recommend papers based on context and keywords and then we let it generate K citation sentences, one for each of the top K recommended papers. Then, we calculate the ROUGE F1 score between the ground truth citation sentence and each of the K sentences and record the highest ROUGE F1 score of them. We compared the “Best-of-top- K ” pipeline to the “generation-only” pipeline, where we directly provide the truly cited paper for citation generation.

We observed that for $K = 5$ and 10, the “Best-of-top- K ” pipeline achieved significantly higher ROUGE scores than the “generation only” pipeline (Table 4), indicating that the paper retrieval module contributes positively to the citation generation process and increases the chance of suggesting appropriate citation sentences. We believe that this result further supports our idea of developing an integrated system for joint retrieval and generation.

3.3 Ablation Study

To analyze the impact of keywords, we evaluated retrieval and generation systems without keywords. For document retrieval, we first prefetch $N_p = 100$ candidates from each corpus and then rank them based on context only. For citation generation, we trained a T5 model to learn to generate citation sentences with only the context and the title and abstract of the cited paper and evaluated it on the

evaluation dataset. We observe a significant degradation in the performance of literature retrieval and citation generation (Table 5), which demonstrates the utility of keywords for recommending relevant papers and generating accurate citations on our platform.

4 Related Work

Recently, AI-driven platforms focused on literature recommendation and scientific paper summarization have been proposed. (keywords: platform, paper: #2) *One such platform is AI Research Navigator (Fadaee et al., 2020), which combines classical keyword search with neural retrieval to discover and organize relevant literature.* (keywords: scientific; summarization; platform, paper #3) *Another platform is Anne O’Tate, which supports user-driven summarization, drill-down and mining of search results from PubMed, the leading search engine for biomedical literature (Smalheiser et al., 2021).* (keywords: related work generation, paper #9) *Chen and Zhuge (2019) automatically generates related work by comparing the main text of the paper being written with the citations of other papers that cite the same references.*

In the previous paragraph, the italicized citation sentences are generated from SCILIT. In generating each sentence, we use all the preceding sentences in the paragraph as contexts and use the keywords in parentheses to obtain the recommended papers and the corresponding citation sentences. The paper index in parentheses indicates the ranked order of recommended papers.

5 Conclusion and Future Work

This paper demonstrates SCILIT, a platform for joint scientific literature retrieval, paper summarization, and citation generation. SCILIT can efficiently recommend papers from hundreds of millions of papers and proactively provides highlights and suggested citations to assist authors in reading and discussing the scientific literature. In addition, our prefetching, reranking, and citation generation system can be conditioned on user-provided keywords, which provides flexibility and adjusts the platform’s response to user intention. In the future, we will further improve the performance of each module, especially the citation generation part, and collect feedback from users to improve the overall workflow and the frontend user experience.

Acknowledgements

We thank the anonymous reviewers for their useful comments.

References

- Titipat Achakulvisut, Daniel Acuna, and Konrad Kornding. 2020. [Pubmed parser: A python parser for pubmed open-access xml subset and medline xml dataset xml dataset](#). *Journal of Open Source Software*, 5(46):1979.
- Shalabh Aggarwal. 2014. *Flask framework cookbook*. Packt Publishing Ltd.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Jingqiang Chen and Hai Zhuge. 2019. [Automatic generation of related work through summarizing citations](#). *Concurrency and Computation: Practice and Experience*, 31(3):e4261. E4261 CPE-16-0462.R2.
- Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. [Structural scaffolds for citation intent classification in scientific publications](#). 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 3586–3596, Minneapolis, Minnesota. Association for Computational Linguistics.
- Holly Else. 2021. Giant, free index to world’s research papers released online. available online at <https://www.nature.com/articles/d41586-021-02895-8>, accessed 2021-12-15. *Nature (Oct 2021)*. <https://doi.org/10.1038/d41586-021-02895-8>.
- Marzieh Fadaee, Olga Gureenkova, Fernando Rejon Barrera, Carsten Schnober, Wouter Weerkamp, and Jakub Zavrel. 2020. [A new neural search and insights platform for navigating and organizing AI research](#). In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 207–213, Online. Association for Computational Linguistics.
- Michael Färber and Adam Jatowt. 2020. Citation recommendation: approaches and datasets. *International Journal on Digital Libraries*, 21(4):375–405.
- Yubin Ge, Ly Dinh, Xiaofeng Liu, Jinsong Su, Ziyao Lu, Ante Wang, and Jana Diesner. 2021. [BACO: A Background Knowledge- and Content-Based Framework for Citing Sentence 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 1466–1478, Online. Association for Computational Linguistics.
- Alexios Gidiotis and Grigorios Tsoumakas. 2020. [A divide-and-conquer approach to the summarization of long documents](#). *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040.
- Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinsh Garg, and Avi Sil. 2020. [Span selection pre-training for question answering](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2782, Online. Association for Computational Linguistics.
- Onur Gökçe, Jonathan Prada, Nikola I. Nikolov, Nianlong Gu, and Richard H.R. Hahnloser. 2020. [Embedding-based scientific literature discovery in a text editor application](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 320–326, Online. Association for Computational Linguistics.
- Nianlong Gu, Elliott Ash, and Richard Hahnloser. 2022a. [MemSum: Extractive summarization of long documents using multi-step episodic Markov decision processes](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6507–6522, Dublin, Ireland. Association for Computational Linguistics.
- Nianlong Gu, Yingqiang Gao, and Richard H. R. Hahnloser. 2022b. [Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking](#). In *Advances in Information Retrieval*, pages 274–288, Cham. Springer International Publishing.
- Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2020. [A deep look into neural ranking models for information retrieval](#). *Information Processing & Management*, 57(6):102067.
- Richard D Hipp. 2000. [Sqlite](https://www.sqlite.org/index.html). <https://www.sqlite.org/index.html>. Version: 3.39.2, Accesses: 2022-07-31.
- Kaggle. 2022. [arXiv Dataset](https://www.kaggle.com/datasets/Cornell-University/arxiv). <https://www.kaggle.com/datasets/Cornell-University/arxiv>. Accesses: 2022-08-01.
- Yang Liu. 2019. [Fine-tune bert for extractive summarization](#).
- Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. [S2ORC: The semantic scholar open research corpus](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

- Zoran Medić and Jan Snajder. 2020. [Improved local citation recommendation based on context enhanced with global information](#). In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 97–103, Online. Association for Computational Linguistics.
- Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero Nogueira dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen McKeown, and Bing Xiang. 2021. [Entity-level factual consistency of abstractive text summarization](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, Online. Association for Computational Linguistics.
- Bethesda (MD): National Library of Medicine. 2003. Pmc open access subset [internet]. <https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/>. Accesses: 2022-07-30.
- Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. [Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features](#). *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. ArXiv: 1703.02507.
- 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.
- Neil R Smalheiser, Dean P Fragnito, and Eric E Tirk. 2021. Anne o’tate: Value-added pubmed search engine for analysis and text mining. *PloS one*, 16(3):e0248335.
- Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. [Asking and Answering Questions to Evaluate the Factual Consistency of Summaries](#). *arXiv:2004.04228 [cs]*. ArXiv: 2004.04228.
- Yifan Wang, Yiping Song, Shuai Li, Chaoran Cheng, Wei Ju, Ming Zhang, and Sheng Wang. 2022. [DisenCite: Graph-Based Disentangled Representation Learning for Context-Specific Citation Generation](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11449–11458.
- Wen Xiao and Giuseppe Carenini. 2019. [Extractive summarization of long documents by combining global and local 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 3011–3021, Hong Kong, China. Association for Computational Linguistics.
- Xinyu Xing, Xiaosheng Fan, and Xiaojun Wan. 2020a. [Automatic Generation of Citation Texts in Scholarly Papers: A Pilot Study](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6181–6190, Online. Association for Computational Linguistics.
- Xinyu Xing, Xiaosheng Fan, and Xiaojun Wan. 2020b. [Automatic generation of citation texts in scholarly papers: A pilot study](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6181–6190, Online. Association for Computational Linguistics.
- Xinnuo Xu, Ondřej Dušek, Jingyi Li, Verena Rieser, and Ioannis Konstas. 2020. [Fact-based content weighting for evaluating abstractive summarisation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5071–5081, Online. Association for Computational Linguistics.
- Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. [Extractive Summarization as Text Matching](#). *arXiv:2004.08795 [cs]*. ArXiv: 2004.08795.

A Hardware Information

We run the backend of SCILIT on a server with dual 64-Core AMD EPYC 7742 2.25GHz Processors, 2TB DDR4 3200MHz ECC Server Memory, and 4×7.68TB NVME GEN4 PM9A3 for storage. The server is also equipped with two nVidia RTX A6000 48GB GPU. The frontend is hosted on Vercel².

B JSON Schema for Database

The details of our unified JSON schema is shown in Listing 1. As metadata we define “Author”, “Title”, “Abstract”, “Venue”, “DOI”, “URL”, and as parsed full text we define “PublicationDate”, “Content.Abstract_Parsed”, and “Content.Fullbody_Parsed”. The parsed abstract or full body contains a list of parsed sections. Each section contains a list of parsed paragraphs, each including a list of parsed sentences. If a sentence cites a paper, we create a “cite span” that records the citation marker such as “[1]”, the position of the citation marker in the sentence, and the cited paper’s index in the “Reference” list.

We implemented a S2ORC parser to convert documents in the S2ORC corpus to our JSON format. For PDFs in the arXiv corpus, we first used the s2orc-doc2json (Lo et al., 2020) to convert them into S2ORC format and then we applied our S2ORC parser. For XML files in the PMCOA corpus, we implemented an XML parser

²<https://vercel.com/>

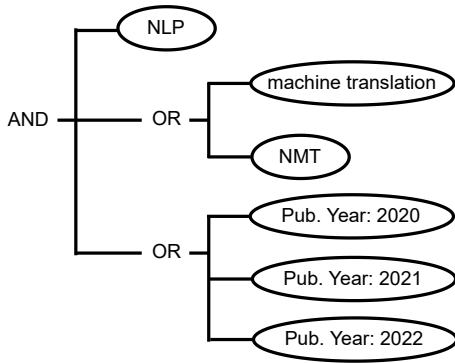


Figure 5: The parsed tree structure of the given keywords string: `NLP; machine learning|NMT; 2020..2022`".

based on Achakulvisut et al. (2020) to convert XML to S2ORC format and then we applied again the S2ORC parser to convert it into our JSON format.

C Prefetching Indexing Implementation

C.1 Inverted Index

The inverted index is a mapping table from keywords (unigrams and bigrams) to paper IDs. We extract keywords from the full text of each document and keep a bigram only if neither word is a stopword. We use `sqlitedict`³ to store the inverted index for each corpus, which is an on-disk hashmap based on an SQLite database that allows us to efficiently obtain the paper ID for a given keyword without loading the entire inverted index into RAM.

Syntax Parsing. Our platform allows users to filter documents using syntax-rich keyword strings. For example, to filter papers that contain the keywords 'NLP' and either 'machine translation' or 'NMT' and that has been published between 2020 and 2022, one can compose a keyword string `NLP; machine learning|NMT; 2020..2022`". We transform this keyword string into a tree of logical operations (Figure 5), wherein each node we denote the logical operations applied to the sub-nodes, and each leaf node contains a keyword. We implemented the tree using a Python dictionary (Listing 2). Then, we recursively traverse all nodes in the tree in a depth-first search, obtain the paper IDs with the keyword in each leaf node, and apply the logical operations indicated in each node to obtain the final paper ID at the root node.

³<https://github.com/RaRe-Technologies/sqlitedict>

C.2 Embedding Index

Structure of the Embedding Index. The embedding index consists of three main components:

The first component is a matrix $\mathcal{M} \in \mathcal{R}^{N \times D}$, where N is the number of documents and D is the dimensionality of document embeddings. Each document's embedding is L2-normalized so that given an L2-normalized query embedding $e_q \in \mathcal{R}^{D \times 1}$, the matrix multiplication $\mathcal{M}e_q \in \mathcal{R}^{N \times 1}$ represents the cosine similarity between the query embedding and all document embeddings. We use $\mathcal{M}e_q$ to rank documents and to obtain the indices of most similar paper embeddings.

The second component is a mapping table from the index of a paper embedding in the matrix \mathcal{M} to the corresponding paper ID in our databases. With this mapping table we can get the papers' content given the top ranked indices during K nearest neighbor search (KNN).

The last component is a reversed mapping table from the paper ID to the corresponding index in the embedding matrix. In our prefetching system, we first use the inverted index to pre-filter a subset of paper IDs based on given keywords. Then we use this reversed mapping table to obtain the corresponding paper embeddings and perform KNN among them.

Multi-Processing Speedup for Brute-Force Nearest Neighbor Search. For a large corpus like S2ORC, the embedding matrix contains up to 136.6 million vectors, and performing matrix multiplication in a single thread is very time-consuming. To take full advantage of the multiple CPU cores on our server, we divide the embedding matrix into 137 shards, each containing about 1 million embeddings. We first run a brute-force nearest neighbor search in parallel to obtain N_p candidates on each shard, and then we rank the $137 \times N_p$ candidates again to obtain the final N_p candidates. Given that our server has 128 cores, we can achieve a nearly linear speedup using multiprocessing KNN with slicing, and mathematically it is equivalent to performing a single KNN over the entire embedding matrix to obtain the closest N_p candidates.

D Joint Retrieval and Citation Generation Examples

We show some specific results of joint paper retrieval and automatic generation of citation sentences. The contexts and keywords we used were obtained from papers in arXiv (Figure 6) and PM-

```

1 {'Author': [{'GivenName': 'Daisuke', 'FamilyName': 'Ida'}, ...],
2 'Title': 'Topology Change of Black Holes',
3 'Abstract': 'The topological structure of the event horizon has been investigated
4 ...',
5 'Venue': '',
6 'DOI': '',
7 'URL': '',
8 'PublicationDate': {'Year': '2007', 'Month': '3'},
9 'Content': {
10   'Abstract': '',
11   'Abstract_Parsed': [{
12     'section_id': '0',
13     'section_title': 'Abstract',
14     'section_text': [{
15       'paragraph_id': '0',
16       'paragraph_text': [{
17         'sentence_id': '0',
18         'sentence_text': 'The topological structure of the event horizon
19         has been investigated in terms of the Morse theory.',
20         'cite_spans': []},
21       # ...
22     ]},
23     # ...
24   ]},
25   'Fullbody': '',
26   'Fullbody_Parsed': [{
27     'section_id': '0',
28     'section_title': 'Introduction',
29     'section_text': [{
30       'paragraph_id': '0',
31       'paragraph_text': [
32         # ...,
33         {
34           'sentence_id': '2',
35           'sentence_text': '[1, 2] This follows from the fact that the total
36           curvature, which is the integral of the intrinsic scalar curvature over the
37           horizon, is positive under the dominant energy condition and from the Gauss-
38           Bonnet theorem.',
39           'cite_spans': [{'start': '4', 'end': '6', 'text': '2'}, {'ref_id': '0
40         '}]},
41         # ...
42       ]},
43     ]},
44   ]},
45   'Reference': [{
46     'Title': 'The large scale structure of space-times',
47     'Author': [{'GivenName': 'S', 'FamilyName': 'Hawking'},
48               {'GivenName': 'G', 'FamilyName': 'Ellis'}],
49     'Venue': '',
50     'PublicationDate': {'Year': '1973'},
51     'ReferenceText': '2. Hawking, S, and G Ellis. "The large scale structure of
52     space-times." (1973).',
53     # ...
54   ]}
55 ]

```

Listing 1: An example of the JSON schema that we used for parsing and storing scientific papers.

```

1 {
2   'operation': 'AND',
3   'elements': [
4     {'operation': 'AND',
5      'elements': [{'operation': None, 'elements': ['nlp']}]},
6     {'operation': 'OR',
7      'elements': [
8        {'operation': 'AND',
9         'elements': [{'operation': None,
10          'elements': ['machine translation']}]},
11        {'operation': 'AND',
12         'elements': [{'operation': None, 'elements': ['nmt']}]}}]},
13    {'operation': 'OR',
14     'elements': [
15       {'operation': None, 'elements': ['publicationdate.year:2020']},
16       {'operation': None, 'elements': ['publicationdate.year:2021']},
17       {'operation': None, 'elements': ['publicationdate.year:2022']}
18     ]
19  ]
20 }

```

Listing 2: The dictionary representation of the tree structure shown in Figure 5.

COA (Figure 7), respectively. In each example, the actual cited paper occurs in the top 5 paper recommendations, which we have highlighted with an underline.

Context:

#OTHERCIT apply a multi-stream CNN model to extract and fuse deep features from the designed complementary shape-motion representations. Zhu et al. #OTHERCIT organize the pairwise displacements between all body joints to obtain a cuboid action representation and use attention-based deep CNN models to focus analysis on actions. Inspired by the fact that the skeleton data is naturally a topological graph, where the joints and bones are regarded as the nodes and edges, Graph Convolutional Network (GCN) is adopted to boost the performance of skeleton based action recognition #OTHERCITS .

Keywords:

skeleton

Recommended Papers and Generated Citations:

UNIK: A Unified Framework for Real-world Skeleton-based Action Recognition

Generated Citation: UNIK #CIT is a generic skeleton-based action recognition model pre-trained on Posetics, a large-scale human skeleton video dataset.

ANUBIS: Skeleton Action Recognition Dataset, Review, and Benchmark

Generated Citation: The ANUBIS dataset #CIT is a large-scale 3D skeleton dataset, which mainly consists of 3D joints, bones, and limbs.

SkeletonNet: Mining Deep Part Features for 3-D Action Recognition

Generated Citation: SkeletonNet #CIT is proposed to extract body-part-based features from each frame of the skeleton sequence, which are translation, rotation, and scale invariant.

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition

Generated Citation: ST-GCN #CIT learns both the spatial and temporal patterns from the skeleton data, which leads to greater expressive power and stronger generalization capability.

Real Citation: ST-GCN #CIT firstly constructs the skeleton as a graph and uses GCN to automatically capture the patterns embedded in the spatial configuration of the joints as well as their temporal dynamics.

SKELETON-BASED ACTION RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

Generated Citation: In #CIT, a novel skeleton transformer module is designed to rearrange and select important skeleton joints automatically.

Figure 6: Example of joint paper retrieval and citation generation. The context text was obtained from arXiv.

Context:

Quadracuspid aortic valve (QAV) is a rare congenital heart disease that is less common than bicuspid and unicuspid aortic valve [#OTHERCITS]. Coronary artery and coronary ostium anomalies are present in 2-10% of patients with QAV [#OTHERCIT, #OTHERCIT, #OTHERCITS]. The functional status of QAV is predominantly pure aortic regurgitation (AR) [#OTHERCIT, #OTHERCIT].

Keywords:

functional status

Recommended Papers and Generated Citations:

ESC Guidelines for the management of grown-up congenital heart disease (new version 2010): The Task Force on the Management of Grown-up Congenital Heart Disease of the European Society of Cardiology (ESC)

Generated Citation: The functional status of QAV is characterized by the presence of aortic regurgitation (AR), aortic stenosis (AR), and aortic stenosis (AR) [#CIT].

Quadracuspid Aortic Valve: A Comprehensive Review

Generated Citation: The clinical manifestations of QAV depend on the functional status of the QAV and the associated disorders [#CIT].

Real Citation: Clinical manifestations, such as palpitations, dyspnea, fatigue, and chest pain, depend on the functional status of QAV and usually present in the fifth or sixth decade of life [#CIT].

A classification system for the bicuspid aortic valve from 304 surgical specimens.

Generated Citation: The functional status of QAV is characterized by the presence of a number of raphe, spatial position of cusps or raphe, and number of raphe [#CIT].

Functional status of the quadracuspid aortic valve/an uncommon coincidence of congenital quadracuspid aortic valve accompanied by hypertrophic obstructive cardiomyopathy.

Generated Citation: The functional status of QAV is characterized by hypertrophic obstructive cardiomyopathy (HOMC) and hypertrophic obstructive cardiomyopathy (HOCM) [#CIT].

Congenital coronary artery fistulae: a rare cause of heart failure in adults

Generated Citation: The functional status of QAV can be improved by epicardial fistula ligation and coronary artery bypass grafting with marked improvement in functional status [#CIT].

Figure 7: Example of joint paper retrieval and citation generation. The context text was obtained from PMCOA.

Massively Multi-Lingual Event Understanding: Extraction, Visualization, and Search

Chris Jenkins, Shantanu Agarwal, Joel Barry, Steven Fincke, Elizabeth Boschee

University of Southern California Information Sciences Institute

{cjenkins, shantanu, joelb, sfincke, boschee}@isi.edu

Abstract

In this paper, we present ISI-CLEAR, a state-of-the-art, cross-lingual, zero-shot event extraction system and accompanying user interface for event visualization & search. Using only English training data, ISI-CLEAR makes global events available on-demand, processing user-supplied text in 100 languages ranging from Afrikaans to Yiddish. We provide multiple event-centric views of extracted events, including both a graphical representation and a document-level summary. We also integrate existing cross-lingual search algorithms with event extraction capabilities to provide cross-lingual event-centric search, allowing English-speaking users to search over events automatically extracted from a corpus of non-English documents, using either English natural language queries (e.g. *cholera outbreaks in Iran*) or structured queries (e.g. find all events of type *Disease-Outbreak* with agent *cholera* and location *Iran*).

1 Introduction

Understanding global events is critical to understanding the world around us—whether those events consist of pandemics, political unrest, natural disasters, or cyber attacks. The breadth of events of possible interest, the speed at which surrounding socio-political event contexts evolve, and the complexities involved in generating representative annotated data all contribute to this challenge. Events are also intrinsically global: many downstream use cases for event extraction involve reporting not just in a few major languages but in a much broader context. The languages of interest for even a fixed task may still shift from day to day, e.g. when a disease emerges in an unexpected location.

The ISI-CLEAR (CROSS-LINGUAL EVENT & ARGUMENT RETRIEVAL) system meets these challenges by building state-of-the-art, language-agnostic event extraction models on top of massively multi-lingual language models. These event

models require only English training data (not even bitext—no machine translation required) and can identify events and the relationships between them in at least a hundred different languages. Unlike more typical benchmark tasks explored for zero-shot cross-lingual transfer—e.g. named entity detection or sentence similarity, as in (Hu et al., 2020)—event extraction is a complex, structured task involving a web of relationships between elements in text.

ISI-CLEAR makes these global events available to users in two complementary ways. First, users can supply their own text in a language of their choice; the system analyzes this text in that native language and provides multiple event-centric views of the data in response. Second, we provide an interface for cross-lingual event-centric search, allowing English-speaking users to search over events automatically extracted from a corpus of non-English documents. This interface allows for both natural language queries (e.g. *statements by Angela Merkel about Ukraine*) or structured queries (*event type = {Arrest, Protest}, location = Iraq*), and builds upon our existing cross-lingual search capabilities, demonstrated in (Boschee et al., 2019).

The primary contributions of this effort are three-fold:

1. Strong, language-agnostic models for a complex suite of tasks, deployed in this demo on a hundred different languages and empirically tested on a representative variety of languages.
2. An event-centric user interface that presents events in intuitive text-based, graphical, or summary forms.
3. Novel integration of cross-lingual search capabilities with zero-shot cross-lingual event extraction.

We provide a video demonstrating the ISI-CLEAR user interface at <https://youtu.be/PE367pyuye8>.

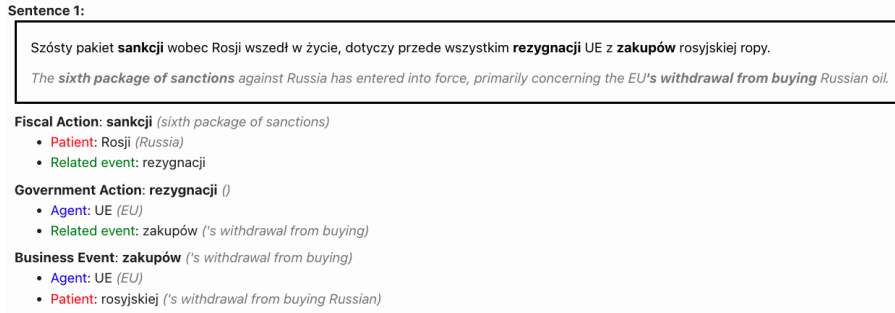


Figure 1: Text-based display of Polish news. The user provides only the Polish text. To aid an English-speaking user, ISI-CLEAR displays the extracted event information not only in Polish but also in English. All processes—including anchor detection, argument extraction, machine translation and span-projection—are carried out in real time.

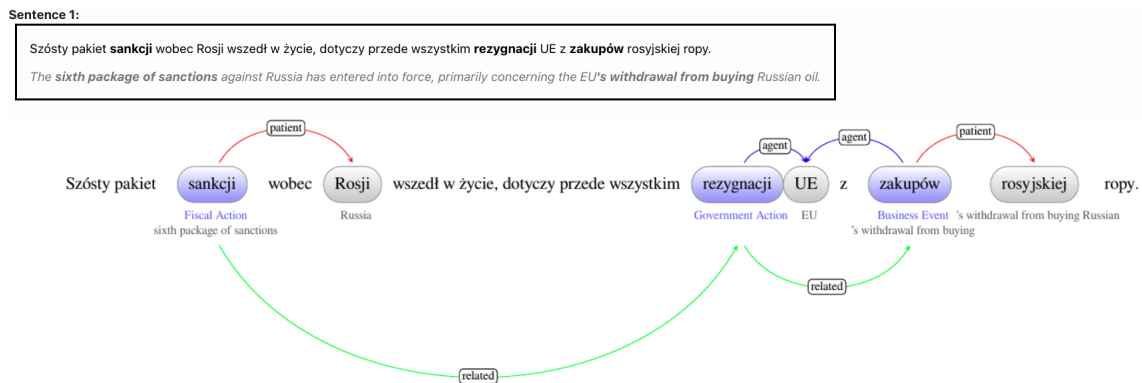


Figure 2: Graph-based display of event information extracted from user provided text in Polish.

2 User Interface

2.1 On-the-Fly Language-Agnostic Event Extraction & Display

In our first mode, users are invited to supply their own text in a language of their choice. The system supports any language present in the underlying multi-lingual language model; for this demo we use XLM-RoBERTa (Conneau et al., 2020), which supports 100 languages ranging from Afrikaans to Yiddish.

After submission, the system displays the results in an initial text-based format, showing the events found in each sentence (Figure 1). For a more intuitive display of the relationships between events, users can select a graphical view (Figure 2). We can easily see from this diagram that the EU is the agent of both the *withdrawal* and the *buying* events, and that the two events are related (the EU is withdrawing from buying Russian oil).

Finally, the user can see an event-centric summary of the document, choosing to highlight either particular categories of event (e.g., *Crime, Military, Money*) or particular participants (e.g., *Ukraine, Putin, Russia*). When one or more categories or

participants are selected, the system will highlight the corresponding events in both the original text and, where possible, in the machine translation. An example of a Farsi document is shown in Figure 3. Here, the system is highlighting three events in the document where Russia is either an agent or a patient of an event. For this demo, we use simple heuristics over English translations to group participant names and descriptions; in future work we plan to incorporate a zero-shot implementation of document co-reference to do this in the original language.

2.2 Cross-Lingual Event-Centric Search

The second mode of the ISI-CLEAR demo allows users to employ English queries to search over events extracted from a foreign language corpus. To enable this, we repurpose our work in cross-lingual document retrieval (Barry et al., 2020) to index and search over event arguments rather than whole documents. A query may specify target *event types* as well as *agent, patient, or location* arguments; it may also include additional words to constrain the *context*. A sample query might ask for *Communicate* events with the agent *Angela Merkel*

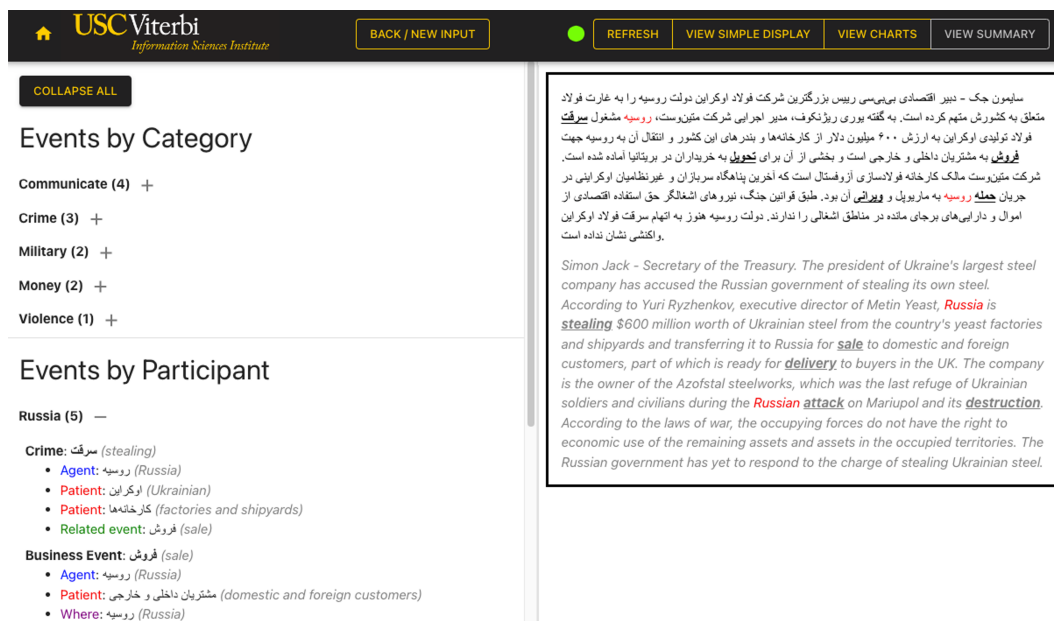


Figure 3: Event-centric summary of Farsi document.

and the context *Ukraine*.

Query specification. We allow queries to be specified in two ways. The first simply asks the user to directly specify the query in structured form: using checkboxes to indicate which event types should be included and directly typing in values for each condition (*agent*, *patient*, etc.). A second and more intuitive method allows users to enter a query as natural language. The system processes the query using the ISI-CLEAR event system and populates a structured query automatically from the results. For instance, if the user enters the phrase *anti-inflation protests in Vietnam*, ISI-CLEAR will detect a *Protest* event with location *Vietnam* in that phrase. It will turn this result into a query with event type *Protest*, location *Vietnam*, and additional context word *anti-inflation*.

Display. We display corpus events in ranked order with respect to the user query. The ranking is a combination of system confidence in the underlying extractions (e.g., is this event *really* located in Vietnam?) and system confidence in the cross-lingual alignment (e.g., is *étudiants internationaux* really a good match for the query phrase *foreign students*?). To estimate the latter, we rely on our prior work in cross-lingual retrieval, where we developed state-of-the-art methods to estimate the likelihood that foreign text f conveys the same meaning as English text e (Barry et al., 2020). We note that for locations, we include containing countries (as determined via Wikidata) in the index so

that a search for *Iran* will return events happening in, e.g., *Tehran*. More specific details on the ranking functions can be found in Appendix A.3.

As part of our display, we break down system confidence by query condition—that is, we separately estimate the system’s confidence in the *agent* vs., say, the *location*. For each condition, we display a “traffic light” indicator that shows the system’s confidence in that condition for an event. Red, yellow, and green indicate increasing levels of confidence; black indicates that there is no evidence for a match on this condition, but that other conditions matched strongly enough for the event to be returned. A sample natural language query and search results are shown in Figure 4.

Corpora. For this demo, we support two corpora: (1) 20,000 Farsi news documents drawn from Common Crawl¹ and (2) ~55K Weibo messages (in Chinese) on the topic of the Russo-Ukrainian crisis (Fung and Ji, 2022).

3 Ontology & Training Data

The ISI-CLEAR demo system is compatible with any event ontology that identifies a set of event types and argument roles. The system expects sentence-level English training data that identifies, for each event, one or more anchor spans and zero or more argument spans (with roles).

For this demonstration, we use the “basic event” ontology and data developed for the IARPA BET-

¹<https://commoncrawl.org/>



Figure 4: Example of search results.

TER program (available at <https://ir.nist.gov/better/>). The ontology consists of 93 event types and a small set of argument roles (*agent*, *patient*, and *related-event*). In other settings, we have trained and tested the underlying system on the publicly available ACE event ontology², showing state-of-the-art zero-shot cross-lingual results in (Fincke et al., 2022). We prefer the BETTER ontology for this demo because of its broad topical coverage and its inclusion of event-event relations (in the form of *related-event* arguments). The ISI-CLEAR system is also designed to attach general-purpose *when* and *where* arguments to any event, regardless of ontology; see section 4.5.

4 System Components

We present here the highlights of our technical approach, which relies on a collection of strong, language-agnostic models to perform all aspects of event extraction and the classification of relationships between events, as well as machine translation and foreign-to-English projection of event output (for display purposes).

4.1 Ingest & Tokenization

Consistent with XLM-RoBERTa, we use Sentence Piece (Kudo and Richardson, 2018) to tokenize text, and at extraction time, our models label each input subword separately. For languages where words are typically surrounded by whitespace, our system then expands spans to the nearest whitespace (or punctuation) to improve overall performance. If the system produces a conflicting sequence of labels for a single word, we apply simple heuristics leveraging label frequency statistics to produce just

²<https://www ldc.upenn.edu/collaborations/past-projects/ace>

one label.

4.2 Anchor Detection

ISI-CLEAR performs anchor identification and classification using a simple beginning-inside-outside (BIO) sequence-labeling architecture composed of a single linear classification layer on top of the transformer stack. For more details please see (Fincke et al., 2022).

4.3 Argument Attachment

For argument attachment, we consider one event anchor A and one role R at a time. We encourage the system to focus on A and R by modifying the input to the language model. For instance, when $A=displaced$ and $R=1$ (*agent*), the input to the language model will be *displaced ; 1 </s> Floods < displaced > thousands last month*. This modification encourages the language model to produce representations of tokens like *thousands* that are contextualized by the anchor and role being examined. The argument attachment model concatenates the language model output vector for each input token with an embedding for event type and applies a linear classifier to generate BIO labels. For more details please see (Fincke et al., 2022).

4.4 Event-Event Relations

ISI-CLEAR can handle arbitrary event-event relations within a sentence, including the special case of event co-reference (when a given event has two or more anchor spans). We consider one event anchor A_1 at a time. Again we modify the input to the language model (by marking A_1 with special characters on either side) to encourage the model to consider all other anchors in light of A_1 . We then represent each event anchor in the sentence (including A_1 itself) as a single vector, generated

by feeding the language model output for its constituent tokens into a bi-LSTM and then concatenating the bi-LSTM’s two final states. (This allows us to smoothly handle multi-word anchors.) To identify the relationship between A_1 and A_2 , if any, we then concatenate the representations for A_1 and A_2 and pass the result to a linear classifier. The final step optimizes over the scores of all such pairwise classifications to label all relations in the sentence.

4.5 When & Where

The ontology used for this demonstration (described in Section 3) does not annotate *when* and *where* arguments. However, these event attributes are critical for downstream utility. We therefore deploy an ontology-agnostic model that can assign dates and locations to events of any type. To do this, we train a question-answering model to answer questions such as *<s> When/Where did the {anchor} happen? </s> Context </s>*. We first train the model on the SQUAD2 dataset (Rajpurkar et al., 2016) and then continue training on the event location and time annotations in the English ACE dataset.

4.6 Machine Translation & Projection

All event extraction happens in the target language; no machine translation (or bitext) is required. However, for system output to be useful to English speakers, translation is highly beneficial. Here, we rely on the 500-to-1 translation engine developed by our collaborators at ISI (Gowda et al., 2021)³. Translation happens after event extraction. We have not optimized this deployment of MT for speed, so we display the results without translation first and then (when the small light in the top toolbar turns green, usually after a few seconds), we can refresh the screen to show results with translations added.

To project anchor and argument spans into machine translation, we require no parallel data for training. Instead, we leverage the fact that the pre-trained XLM-RoBERTa embeddings are well aligned across languages and have been shown to be effective for word alignment tasks (Dou and Neubig, 2021). The similarity of a word in a foreign-language sentence to a word in the parallel English sentence is determined by the cosine distance between the embeddings of the two words. We leverage the Itermax algorithm (Jalili Sabet et al., 2020) to find the best phrase matches. Since

³Available at <http://rtg.isi.edu/many-eng/>.

we avoid making any bespoke language specific decisions, our projection technique is highly scalable and can project from any of the 100 languages on which XLM-RoBERTa was pre-trained on.

5 System Evaluation & Analysis

We evaluate our system on a variety of languages and ontologies and compare where possible to existing baselines. Following community practice, e.g. Zhang et al. (2019), we consider an anchor correct if its offsets and event type are correct, and we consider an argument correct if its offsets, event type, and role find a match in the ground truth. For event coreference (same-sentence only), we consider each anchor pair separately to produce an overall F-score.

Table 1 provides overall scores in several settings where multi-lingual event annotations are available. All models are trained on English data only. For the ACE data, we follow (Huang et al., 2022). The BETTER Basic task is described in Section 3; there are two ontologies (Basic-1 and Basic-2) from different phases of the originating program. The BETTER Abstract task is similar to BETTER Basic, but all action-like phrases are annotated as events, with no further event type specified⁴; valid roles are only *agent* and *patient* (McKinnon and Rubino, 2022). More dataset statistics are found in Appendix A.1.

It is difficult to compare system accuracy across languages; a lower score in one language may reflect a real difference in performance across languages—or just that one set of documents is harder than another. Still, we observe the following. First, performance on anchors seems most sensitive to language choice—for instance, we note that Arabic and Chinese anchor performance on ACE differs by almost 10 points. For arguments, however, non-English performance is relatively consistent given a task—but varies more widely between tasks. Second, we note that cross-lingual performance seems best on anchors, where it exceeds 80% of English performance for all but one condition. In contrast, argument performance varies more widely, with many conditions below 70% of English (though some as high as 89%).

We also compare against existing published baselines where possible. There are relatively few published results on cross-lingual event anchor detection (and none that we could find on the task of

⁴Since abstract events lack event types, we also require anchor offsets to match when scoring arguments.

Task	ACE			Basic-1		Basic-2		Abstract			
	en	ar	zh	en	ar	en	fa	en	ar	fa	ko
Anchors	71.2	58.1	49.6	64.2	52.5	64.6	54.3	87.4	78.3	72.5	78.9
Arguments	72.1	51.5	51.7	64.5	51.5	71.6	64.0	69.8	45.0	45.7	45.0
Event coreference	–	–	–	83.4	67.9	86.5	65.9	–	–	–	–

Table 1: Component-level accuracy by language / task. Dataset statistics are available in Appendix A.1. ACE lacks same-sentence event coreference so those figures are omitted. Event coreference is peripheral to the overall Abstract task; we chose to not model it explicitly and exclude it here.

cross-lingual event co-reference as defined here). To benchmark performance on anchors, we turn to MINION (Pouan Ben Veyseh et al., 2022), a multi-lingual anchor-only dataset that uses a derivative of the ACE ontology. For a fair comparison, we retrained our model (tuned for use with XLM-RoBERTa large) with XLM-RoBERTa base; we did not adjust any hyperparameters. Table 2 shows that the ISI-CLEAR model performs on average 2.7 points better than the reported MINION numbers for cross-lingual settings. We also show the numbers from our actual demo models (trained with XLM-RoBERTa large) for comparison.

	base			large
	MINION	ISI-CLEAR	Δ	ISI-CLEAR
en	79.5	78.9	-0.6	78.0
es	62.8	62.3	-0.5	65.3
pt	72.8	71.1	-1.7	75.0
pl	60.1	52.6	-7.5	66.4
tr	47.2	52.0	+4.8	56.5
hi	58.2	72.2	+14.0	72.7
ko	56.8	64.1	+7.3	63.5
AVG	59.7	62.4	+2.7	66.6

Table 2: Cross-lingual anchor detection (F1) for MINION dataset, training on English only. Average is across all cross-lingual settings.

For argument detection, much more published work exists, and we show in Table 3 that ISI-CLEAR achieves state-of-the-art performance on all ACE datasets, comparing against the previous state-of-the-art as reported in Huang et al. (2022).

6 Related Work

Several recent demos have presented multi-lingual event extraction in some form, but most assume training data in each target language (e.g. Li et al.

	X-GEAR	ISI-CLEAR
en	71.2	72.1
ar	44.8	51.5
zh	51.5	51.7

Table 3: Cross-lingual argument detection (F1) for ACE over gold anchors, training on English only.

(2019) or Li et al. (2020)) or translate foreign-language text into English before processing (e.g. Li et al. (2022)). In contrast, the focus of our demo is making events available in languages for which no training data exists. Other demos have shown the potential of zero-shot cross-lingual transfer, but on unrelated tasks, e.g. offensive content filtering (Pelicon et al., 2021). Akbik et al. (2016) uses annotation projection from English FrameNet to build target-language models for frame prediction; the focus of the demo is then on building effective queries over language-agnostic frame semantics for extraction. Finally, Xia et al. (2021) also produce FrameNet frames cross-lingually (using XLM-RoBERTa), but in contrast to our work, several of their supporting models use target-language data, and they also supply only a simpler user interface and lack the cross-lingual search-by-query capability that is a key aspect of our demo.

7 Conclusion

ISI-CLEAR provides a monolingual English-speaking user with effective access to global events, both on-demand (extracting events from input of a user’s choice) or as a set of indexed documents accessible via cross-lingual search. The system provides a variety of visualizations and modes for engaging with system results. We look forward to future work improving the quality of the underlying components and exploring additional capabilities to cross language barriers and expand access to

information around the globe.

Limitations

Our core approach is limited by the underlying multi-lingual language model it employs. For this demo, we are therefore limited to the 100 languages that make up the XLM-RoBERTa training set. Performance also varies across languages, tracking in part (though not in whole) with the volume of training data available for each language when building the multi-lingual language model. For instance, anecdotally, the performance on Yiddish (34M tokens in the CC-100 corpus used to train XLM-RoBERTa) is inferior to that of Farsi (13259M tokens). We have provided empirical results for eleven languages and five tasks, but it would be ideal to have a broader set of test conditions; unfortunately, annotated datasets for events are much less common than for simpler tasks like named entity recognition.

A second limitation of our system involves compute requirements. We employ multiple separate components for event extraction (e.g., for anchor detection vs. argument attachment), which increases memory/GPU footprint compared to a more unified system.

Finally, our system assumes an existing ontology and (English) training data set; it would be interesting to explore zero-shot ontology expansion in future work.

Ethics Statement

One important note is that our system is designed to extract information about events that are reported in text, with no judgment about their validity. This can lead a user to draw false conclusions. For instance, the system might return many results for a person X as the agent of a *Corruption* event, but this does not necessarily mean that X is actually corrupt. This should be prominently noted in any use case for this demonstration system or the underlying technologies.

Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the

official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

- Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yonas Kbrom, Yunyao Li, and Huaiyu Zhu. 2016. [Multilingual information extraction with PolyglotIE](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 268–272, Osaka, Japan. The COLING 2016 Organizing Committee.
- Joel Barry, Elizabeth Boschee, Marjorie Freedman, and Scott Miller. 2020. [SEARCHER: Shared embedding architecture for effective retrieval](#). In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, pages 22–25, Marseille, France. European Language Resources Association.
- Elizabeth Boschee, Joel Barry, Jayadev Billa, Marjorie Freedman, Thamme Gowda, Constantine Lignos, Chester Palen-Michel, Michael Pust, Banriskhem Kayang Khonglah, Srikanth Madikeri, Jonathan May, and Scott Miller. 2019. [SARAL: A low-resource cross-lingual domain-focused information retrieval system for effective rapid document triage](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 19–24, Florence, Italy. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Zi-Yi Dou and Graham Neubig. 2021. [Word alignment by fine-tuning embeddings on parallel corpora](#). *CoRR*, abs/2101.08231.
- Steven Fincke, Shantanu Agarwal, Scott Miller, and Elizabeth Boschee. 2022. Language model priming for cross-lingual event extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Yi R. Fung and Heng Ji. 2022. [A weibo dataset for the 2022 russo-ukrainian crisis](#).
- Thamme Gowda, Zhao Zhang, Chris Mattmann, and Jonathan May. 2021. [Many-to-English machine translation tools, data, and pretrained models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

- International Joint Conference on Natural Language Processing: System Demonstrations*, pages 306–316, Online. Association for Computational Linguistics.
- 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 *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.
- Kuan-Hao Huang, I-Hung Hsu, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. **Multilingual generative language models for zero-shot cross-lingual event argument extraction**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4633–4646, Dublin, Ireland. Association for Computational Linguistics.
- Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. **SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, 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**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Manling Li, Revanth Gangi Reddy, Ziqi Wang, Yishyuan Chiang, Tuan Lai, Pengfei Yu, Zixuan Zhang, and Heng Ji. 2022. **COVID-19 claim radar: A structured claim extraction and tracking system**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 135–144, Dublin, Ireland. Association for Computational Linguistics.
- Manling Li, Ying Lin, Joseph Hoover, Spencer Whitehead, Clare Voss, Morteza Dehghani, and Heng Ji. 2019. **Multilingual entity, relation, event and human value extraction**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 110–115, Minneapolis, Minnesota. Association for Computational Linguistics.
- Manling Li, Alireza Zareian, Ying Lin, Xiaoman Pan, Spencer Whitehead, Brian Chen, Bo Wu, Heng Ji, Shih-Fu Chang, Clare Voss, Daniel Napierski, and Marjorie Freedman. 2020. **GAIA: A fine-grained multimedia knowledge extraction system**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 77–86, Online. Association for Computational Linguistics.
- Timothy McKinnon and Carl Rubino. 2022. The IARPA BETTER program abstract task four new semantically annotated corpora from IARPA’s BETTER program. In *Proceedings of The 14th Language Resources and Evaluation Conference*.
- Andraž Pelicon, Ravi Shekhar, Matej Martinc, Blaž Škrj, Matthew Purver, and Senja Pollak. 2021. **Zero-shot cross-lingual content filtering: Offensive language and hate speech detection**. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, pages 30–34, Online. Association for Computational Linguistics.
- Amir Poursan Ben Veyseh, Minh Van Nguyen, Franck Dernoncourt, and Thien Nguyen. 2022. **MINION: a large-scale and diverse dataset for multilingual event detection**. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2286–2299, Seattle, United States. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. **SQuAD: 100,000+ Questions for Machine Comprehension of Text**. *arXiv e-prints*, page arXiv:1606.05250.
- Milan Straka. 2018. **UDPipe 2.0 prototype at CoNLL 2018 UD shared task**. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.
- Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. 2021. **LOME: Large ontology multilingual extraction**. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 149–159, Online. Association for Computational Linguistics.
- Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1:99–120.

A Appendix

A.1 Dataset Statistics

We report results for a variety of different tasks in a variety of different languages. We outline the sizes for these diverse datasets in Tables 4 and 5. The tasks use five different ontologies; we also report the number of event types for each ontology in Table 6.

A.2 Speed

Table 7 presents speed results for six representative languages, calculated as number of seconds per

	Train		Development	
	# Characters	# Events	# Characters	# Events
ACE	1,335,035	4,202	95,241	450
Basic-1	171,267	2,743	35,590	560
Basic-2	419,642	5,995	87,425	1,214
Abstract	557,343	12,390	67,266	1,499
MINION	4,388,701	14,189	544,758	1,688

Table 4: Size of English training and development sets in number of documents and number of events.

	Lang.	# Characters	# Events
ACE	en	104,609	403
	ar	44,003	198
	zh	22,452	189
Basic-1	en	33,169	569
	ar	238,133	5,172
Basic-2	en	82,296	1,139
	fa	639,6951	11,559
Abstract	en	68,863	1,527
	ar	189,174	5,339
	fa	607,429	15,005
	ko	327,811	16,704
MINION	en	554,680	1,763
	es	161,159	603
	pt	73,610	200
	pl	197,270	1,234
	tr	175,823	814
	hi	57,453	151
	ko	332,023	164

Table 5: Size of test sets in number of documents and number of events.

100 “words”. For this exercise we consider words to be the output of UDPipe’s language-specific tokenization (Straka, 2018). The primary driver of speed difference is that, given XLM-RoBERTa’s fixed vocabulary, different languages will split into more or fewer subwords on average. For instance, an average Korean word will produce at least half again as many subwords than, say, an average Farsi word; this is presumably why 100 words of Korean takes about 70% longer to process than 100 words of Farsi. On average, for a standard short news article (200 words), we expect to wait about two seconds for extraction and an additional six or seven seconds for MT and projection. We did not optimize our selection of MT package for speed

Ontology	# of Event Types
ACE	33
Basic-1	69
Basic-2	93
Abstract	1
MINION	16

Table 6: Number of event types in each ontology.

(e.g., it decodes one sentence at a time instead of batching); this could easily be updated in future work to be more efficient.

	en	ar	fa	ko	ru	zh
Event	1.1	1.0	0.9	1.5	0.8	1.1
Display	n/a	2.6	2.8	4.1	3.4	3.9

Table 7: Processing speed (seconds per 100 words). Event processing includes ingest, tokenization, anchors, arguments, event-event relationships, and when/where extraction. Display processing includes components solely required for display (MT and projection). We use 11GB GTX 1080Ti GPUs for extraction/projection and use a 48GB Quadro RTX 8000 GPU for MT.

A.3 Search Ranking

ISI-CLEAR extracts a large number of events from the documents indexed from search, some of which vary in quality and some of which will match more or less confidently to an English query. The ranking function described here significantly improves the usability of our search results.

The goal of our search ranking function is to rank each extracted event E with respect to a user query Q . To calculate $score(Q, E)$, we combine two separate dimensions of system confidence:

1. *Cross-lingual alignment confidence (CAC)*: are the components of E reasonable translations of the query terms? For instance, is *étu-*

ditants internationaux a good match for the query phrase *foreign students*? Here, we assume the existence of a cross-lingual retrieval method $cac(e, f)$ that estimates the likelihood that foreign text f conveys the same meaning as English text e , as in our prior work (Barry et al., 2020).

2. *Extraction confidence (EC)*: how likely is it that the elements of E were correctly extracted in the first place? Here we use confidence measures (denoted ec) produced by individual system components.

To combine these dimensions, we consider each query condition separately (summing the results). For simplicity we describe the scoring function for the *agent* condition:

$$\begin{aligned} score(Q_{agent}, E_{agent}) = & \\ & \beta * ec(E_{agent}) * cac(Q_{agent}, E_{agent}) + \\ & (1 - \beta) * cac(Q_{agent}, E_{sentence}) \end{aligned}$$

The first term of this equation captures the two dimensions described above. The second term allows us to account for agents missed by the system, letting us give “partial credit” when the user’s search term is at least found in the nearby context (e.g., in $E_{sentence}$). Based on empirical observation, we set β to 0.75.

We follow the same formula for *patient* and *location*. For *context* we use only the final term $cac(Q_{topic}, E_{sentence})$ since *context* does not directly correspond to an event argument.

For now, event type operates as a filter with no score attached; in future work we will incorporate both the system’s confidence in the event type as well as a fuzzy match over nearby event types (e.g., allowing for confusion between *Indict* and *Convict*).

YANMTT: Yet Another Neural Machine Translation Toolkit

Raj Dabre¹ Diptesh Kanojia³
Chinmay Sawant² Eiichiro Sumita⁴

National Institute of Information and Communications Technology^{1,4}

Surrey Institute for People-centred AI³

University of Surrey^{2,3}

¹raj.dabre@nict.go.jp ²chinmayssawant44@gmail.com
³d.kanojia@surrey.ac.uk ⁴eiichiro.sumita@nict.go.jp

Abstract

In this paper, we present our open-source neural machine translation (NMT) toolkit called “Yet Another Neural Machine Translation Toolkit” abbreviated as YANMTT¹ which is built on top of the HuggingFace Transformers library. YANMTT aims to enable pre-training and fine-tuning of sequence-to-sequence models with ease. It can be used for training parameter-heavy models with minimal parameter sharing and efficient, lightweight models via heavy parameter sharing. Additionally, efficient fine-tuning can be done via fine-grained tuning parameter selection, adapter and prompt tuning. Our toolkit also comes with a user interface that can be used to demonstrate these models and visualize the attention and embedding representations. Apart from these core features, our toolkit also provides other advanced functionalities such as but not limited to document/multi-source NMT, simultaneous NMT, mixtures-of-experts and model compression.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2015) is an end-to-end machine translation (MT) approach known for obtaining state-of-the-art results for a variety of language pairs. Thanks to publicly available toolkits such as OpenNMT, Fairseq, Tensor2tensor, etc.; For NMT, and Natural Language Generation (NLG) in general, model training has become easier. Additionally, fine-tuning of large pre-trained models such as mBART (Liu et al., 2020) and mT5 (Xue et al., 2021) have led to significant advances for low-resource languages. Recently, the Transformers library from HuggingFace has made fine-tuning an accessible option. Parameter-efficient fine-tuning via adapters and prompts has recently gained popularity. This has led to the development of

AdapterHub, which allows people to use pre-trained adapters for many tasks or easily train their custom adapters.

However, when it comes to pre-training sequence-to-sequence models from scratch, we noticed that there is very little support². This is, presumably, because pre-training is computationally intensive, especially when the number of parameters is large, and not everyone can do this. However, not all situations need large models, and with the advancements in low-precision training and parameter-efficient architectures, we feel that democratizing pre-training is necessary. Additionally, relying on separate libraries for pre-training and fine-tuning can be quite exhausting. To this end, we decided to develop a publicly available toolkit to bring both into one place while simultaneously focusing on parameter efficiency.

We call our toolkit **YANMTT** which stands for “Yet Another Neural Machine Translation Toolkit” which is built on top of the Transformers library. YANMTT relies on a substantially modified version of the mBART model’s code and contains simple scripts for NMT pre-training and fine-tuning. Our modifications revolve around parameter efficiency by implementing heavy parameter sharing and incorporating adapters and prompts into the model. In order to enable users to better understand, and modify if needed, the flow of pre-training and fine-tuning, we heavily annotate our code with comments. YANMTT also comes with a user interface that can be used to demo any developed models as well as analyze them by visualizing their attentions and embedding representations. We hope that YANMTT will help entice more researchers into advancing the field of efficient sequence-to-sequence pre-training and fine-tuning. While the main focus is on NMT, readers should note that this toolkit can also be used for general purpose NLG.

¹<https://github.com/prajdabre/yanmtt>

²At the time of creating this toolkit in 2021, there was no easily available script for sequence-to-sequence pre-training.

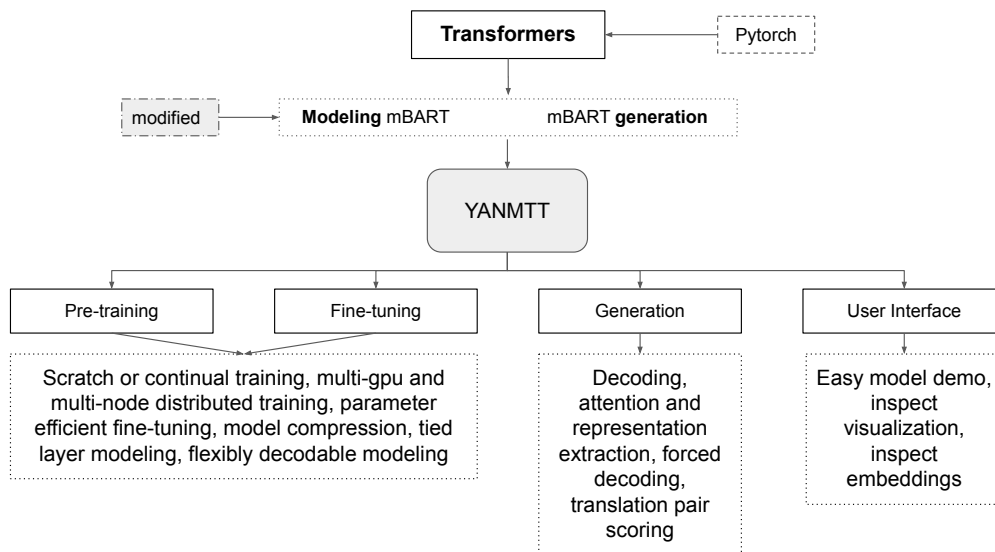


Figure 1: An architecture overview of the YANMTT.

2 Related Work

Tensor2tensor³ is a deprecated library for training recurrent, convolutional as well as transformer models for a variety of sequence-to-sequence applications. It has been replaced by Trax⁴. While Tensor2tensor uses TensorFlow as a backend, Fairseq⁵ is based on PyTorch and it also allows one to train a variety of NMT models. Unlike Tensor2tensor, Fairseq contains all necessary functionality for pre-training NMT models, but there is a severe lack of instructions for the same. OpenNMT (Klein et al., 2017), originally developed for recurrent NMT models, is based on TensorFlow as well as PyTorch. THUMT⁶ is an NMT training toolkit based on TensorFlow, PyTorch and Theano.

Most recently, the Transformers (Wolf et al., 2020) library by HuggingFace, based on PyTorch and TensorFlow has become popular as it allows users to share trained models easily. In Transformers, the instructions for fine-tuning pre-trained models are abundant, but at the time of YANMTT’s development (early 2021), there was no complete script for pre-training. On the HuggingFace hub, the central repository for all models trained with Transformers, it is possible to load and run models, but enabling users to locally demo and inspect their

own models is also important from the perspective of privacy. Finally, for parameter-efficient fine-tuning, AdapterHub (Pfeiffer et al., 2020) builds on top of Transformers and enables users to leverage existing or custom-trained adapters.

All of the above toolkits and libraries are invaluable, but none appear to be a complete solution for sequence-to-sequence pre-training, parameter efficient fine-tuning and model demoing and inspection, a gap which YANMTT aims to fill.

3 The Toolkit: YANMTT

YANMTT, Yet Another Neural Machine Translation Toolkit, relies on the Transformers library and uses PyTorch. We use only the mBART implementation (for now) from Transformers and write several wrapper scripts enabling multilingual sequence-to-sequence pre-training, parameter efficient fine-tuning, decoding and attention and representation extraction. To enable users to quickly demonstrate and visually inspect trained models, we provide a user interface. We also modify the mBART modelling code to provide several advanced features. We provide the modified code along with our toolkit. We also provide example data and usage instructions in the form of example scripts. We encourage the reader to look at our toolkit⁷ and watch the demo video⁸. Figure 1 contains an overview of YANMTT architecture.

³<https://github.com/tensorflow/tensor2tensor>

⁴<https://github.com/google/trax>

⁵<https://github.com/facebookresearch/fairseq>

⁶<https://github.com/THUNLP-MT/THUMT>

⁷<https://github.com/prajdabre/yanmtt>

⁸<https://youtu.be/ee38gda5qnc>

```

Saving the model
Loading from checkpoint
124000 2.27 43.59 seconds for 100 batches. Memory used post forward / backward passes: 9.13 / 5.27 GB.
124100 2.26 208.63 seconds for 100 batches. Memory used post forward / backward passes: 9.13 / 5.22 GB.
124200 2.25 209.06 seconds for 100 batches. Memory used post forward / backward passes: 9.02 / 5.05 GB.
124300 2.3 205.34 seconds for 100 batches. Memory used post forward / backward passes: 7.92 / 4.85 GB.
124400 2.25 203.92 seconds for 100 batches. Memory used post forward / backward passes: 8.6 / 5.13 GB.
124500 2.3 211.39 seconds for 100 batches. Memory used post forward / backward passes: 8.59 / 4.98 GB.
124600 2.3 210.7 seconds for 100 batches. Memory used post forward / backward passes: 9.49 / 5.24 GB.
124700 2.32 209.74 seconds for 100 batches. Memory used post forward / backward passes: 8.87 / 5.25 GB.
124800 2.35 206.13 seconds for 100 batches. Memory used post forward / backward passes: 8.61 / 4.9 GB.
124900 2.26 204.53 seconds for 100 batches. Memory used post forward / backward passes: 9.23 / 5.26 GB.
Saving the model after the final step
The best BLEU using SacreBLEU was: 31.06
The corresponding step was: 49000
~/extStorage/akshith/yanmtt on main !12 ?33 ..... took 1d 13h 34m 45s Py

```

Figure 2: Screenshot obtained after fine-tuning mBART-50 for English-Telugu NMT where the model training was performed over 1 x A40 GPU over ~1 day and 13 hours.

3.1 Training

YANMTT enables multilingual training of sequence-to-sequence models; while also supporting full or mixed-precision training on a single GPU or on multiple GPUs⁹ across machines.

Tokenization: While mBART uses BPE, mT5 uses *sentencepiece* for subword segmentation. YANMTT allows users to train and use both types of subword segmenters.

Pre-training: We currently support mBART style text-infilling and sentence shuffling, as well as mT5 and MASS style masked span-prediction for pre-training from scratch. Note that sentence shuffling is optional since it is useful only when the pre-training data is in the form of documents. We also support continued pre-training of existing mBART-25, mBART-50, BART and IndicBART (Dabre et al., 2022) models¹⁰. This should enable users to adapt pre-trained models using monolingual corpora on a downstream NLG task such as NMT.

Fine-tuning: Users may train their own sequence-to-sequence models from scratch or fine-tune pre-trained models. Fine-tuning can be done on the user’s own models or official pre-trained models like mBART-25, mBART-50, BART and IndicBART. Users have fine-grained control over what parts of the pre-trained models they want to use for partial initialization. This way, the fine-tuned¹¹ models can be shrunk or grown as required.

⁹We directly use the distributed data-parallel functionality of PyTorch instead of the Accelerate library so users better can understand and control how distribution is done.

¹⁰Any models based on the BART or mBART architecture will work. We plan to support the adaptation of other sequence-to-sequence models soon.

¹¹not just for fine-tuning on a downstream task, but also when doing continued pre-training. Additionally, this functionality may be used if one wishes to expand the model’s capacity step by step.

Logging: Figure 2 shows the logs generated during model training depicting model loss, evaluation set scores, timings and memory consumption. Although we do not show it here, this, along with gradient information, is also logged on TensorBoard.

3.2 Parameter Efficiency

Using YANMTT, parameter efficiency can be achieved when fine-tuning or training from scratch.

Lightweight Fine-Tuning: We enable fine-tuning of adapters and prompts to enable users to fine-tune minimal number of parameters. We implement various adapters mentioned in He et al. (2022) such as Houlsby adapter, FFN-only adapter, parallel adapter, hypercomplex adapter (Le et al., 2021) and IA³ adapter (Liu et al., 2022). Regarding prompts, currently prefix tuning is supported. It is also possible to mix-and-match adapters and prompts. Users may also specify a list of parameters which they want to fine-tune for more control over the process.

Parameter Sharing: When training models from scratch, we enable recurrently stacked layers (Dabre and Fujita, 2019) involving tying parameters of layers and the resultant models, which we call ALBERT, tend to be 50-70% smaller.

3.3 Decoding

Users can decode the models, extract encoder or decoder representations, generate heatmaps of attentions and score translation pairs.

3.4 User Interface (UI)

To enable users to locally test or demo their models and inspect them, we provide a web-based UI based on Flask. This interface can be hosted online by using software such as ngrok¹² or an Apache server.

¹²<https://ngrok.com>

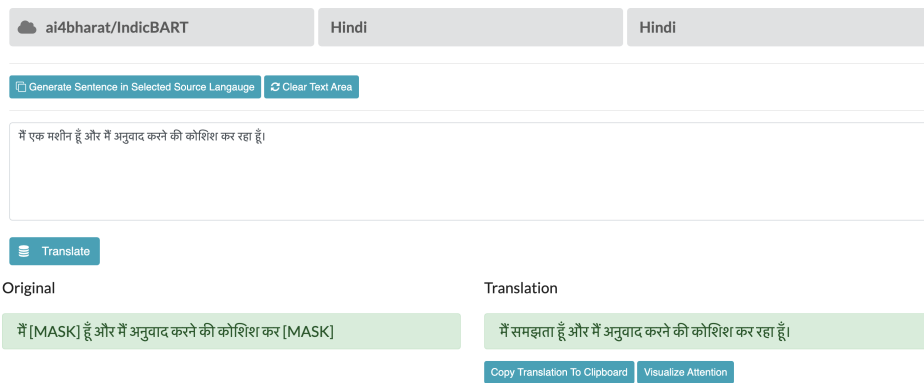


Figure 3: YANMTT User Interface for inspecting the IndicBART model by masking random words from the input sentence in the Hindi language. IndicBART was trained using YANMTT.

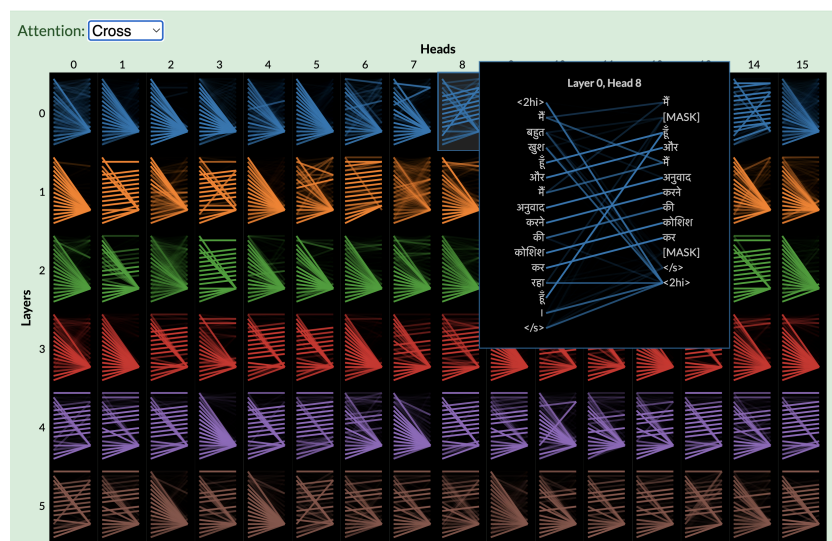


Figure 4: Visualization of cross-attention, where masked words are predicted for a sentence in Hindi.

Model Decoding: Using the GUI, the user may load models and decode sentences. Models supported are those trained locally or official based on the mBART code. If the model to be used is a denoising model, then users may use decoding for MASK prediction as in Figure 3. If the model to be used was fine-tuned for a downstream task such as NMT, then it can be used for generating relevant outputs for that task input.

Attention Visualization: To enable users to inspect their model attentions, we use the bertviz¹³ library to visualize multi-head, self- or cross-attention across all layers in the encoder and decoder. As shown in Figure 4, users can select the type of attention, the desired layer and attention head and get more detailed information.

Representation Visualization: We also integrate TensorBoard embedding projector visualization

into our user interface to visualize the model layer representations¹⁴ as shown in Figure 5. This can be especially useful for understanding model representations in multilingual settings.

3.5 Advanced Features

In addition to the core features above, we also enable YANMTT users to perform the following:

Model Compression: We enable users to compress models by distillation (Kim and Rush, 2016) of models using either or all of 3 different ways: teacher-student cross-entropy minimization, minimizing the mean squared difference between the hidden layer representations of the teachers and students and minimizing the cross entropy between the self/cross-attentions of the teacher and students.

¹⁴For now, we support the encoder’s final layer’s average representation. Additional flexibility in terms of layer choice as well as decoder representations will be provided in future versions of YANMTT.

¹³<https://github.com/jessevig/bertviz>

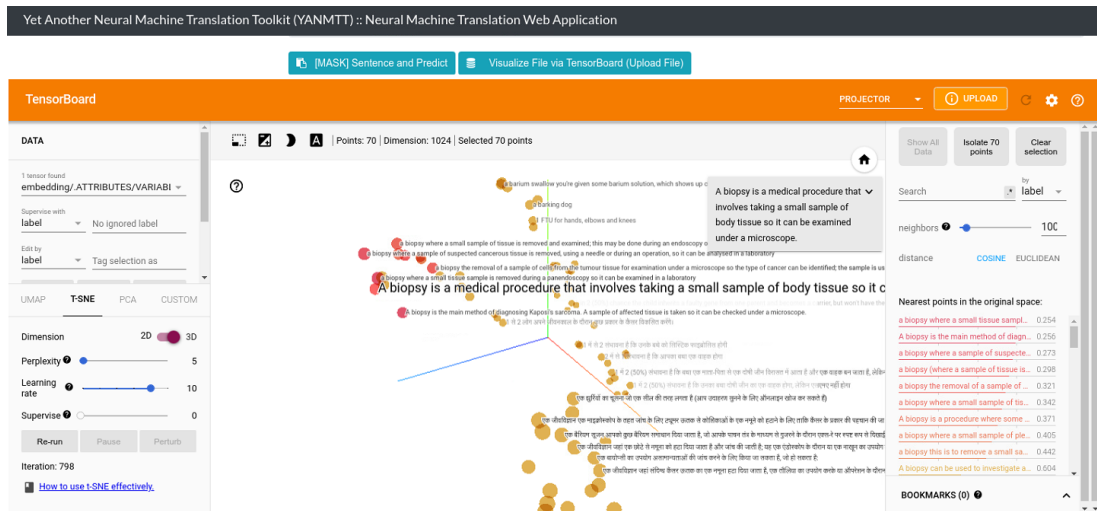


Figure 5: Sentence representations visualization via TensorBoard integrated into the YANMTT user interface.

Mixtures-of-Experts: Mixtures-of-expert (MoE) layers can replace standard feed-forward layers and enable the user to train large models with a billion parameters or more. Model parallel training is currently unavailable, so the largest trainable model is limited by computing resources available¹⁵. A similar approach was used by Facebook to train the largest known multilingual NMT model supporting over 200 language pairs (Costa-jussà et al., 2022). **Document and Multi-source Translation:** Pre-trained models often use document-level data, and we considered it prudent to support explicit document-level translation approaches. Since document context is often treated as an additional input, our document translation implementation can be used for multi-source MT (Dabre et al., 2017). **Simultaneous Translation:** In a real-time setting, simultaneous MT can be useful. To enable users to test their models in simultaneous MT settings, we implement ‘wait-k’ training and decoding (Ma et al., 2019) which can also be combined with document and multi-source MT. Apart from this, there are several **other features which are listed on the YANMTT’s GitHub repository.**

3.6 YANMTT Adoption & Future Plans

YANMTT has 93 stars on GitHub, indicating that it is being noticed and used. In particular, IndicBART¹⁶, its variants and fine-tuned version (Dabre et al., 2022) were developed with YANMTT and have seen around 8500 downloads thus

¹⁵With A100 80 GB GPUs, it is possible to train models with 10-20 billion parameters given reasonable batch size.

¹⁶<https://huggingface.co/ai4bharat/IndicBART>

far from HuggingFace hub. We have the following future plans for our toolkit:

1. Supporting additional pre-training approaches like PEGASUS and CSP.
2. Low-precision model parallel training for better scaling of large models.
3. Comprehensive support for all types of adapters and prompt tuning.
4. An improved user interface to enable better visualization of model internals.
5. Integration of existing post-hoc model explainability techniques.

4 Conclusion

We have presented our open-source toolkit called "Yet Another Neural Machine Translation Toolkit", also known as YANMTT. YANMTT allows users to pre-train and fine-tune their own multilingual sequence to sequence models. Our toolkit can be used for training models at a reasonable scale, as well as to perform parameter efficient fine-tuning via adapters and prompts. We provide a convenient user interface for model demonstration and inspection, as well as the ability to visualize attention and model representations. We also implemented functionalities for compressing large models via selective parameter transfer and knowledge distillation approaches. Additionally, we have provided basic functionalities for simultaneous and document/multi-source NMT. YANMTT appears to be modestly adopted by researchers, and we plan to further specialize it for better at-scale training and efficient fine-tuning.

5 Limitations

Although YANMTT can be used to train models at scale, the lack of model-parallel training limits the size of models to those that can fit on single GPUs. YANMTT tokenizes sentences on the fly and does not pre-process them, so the overall training speed is slightly slower than that of Fairseq or Tensor2Tensor; however, we plan to fix this soon.

6 Acknowledgements

A part of this work was conducted under the commissioned research program "Research and Development of Advanced Multilingual Translation Technology" in the "R&D Project for Information and Communications Technology (JPMI00316)" of the Ministry of Internal Affairs and Communications (MIC), Japan.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. [No language left behind: Scaling human-centered machine translation](#). *arXiv preprint arXiv:2207.04672*.
- Raj Dabre, Fabien Cromieres, and Sadao Kurohashi. 2017. [Enabling multi-source neural machine translation by concatenating source sentences in multiple languages](#). In *Proceedings of MT Summit XVI, vol.1: Research Track*, pages 96–106, Nagoya, Japan.
- Raj Dabre and Atsushi Fujita. 2019. [Recurrent stacking of layers for compact neural machine translation models](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6292–6299.
- Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh Khapra, and Pratyush Kumar. 2022. [IndicBART: A pre-trained model for indic natural language generation](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1849–1863, Dublin, Ireland. 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](#). In *International Conference on Learning Representations*.
- Yoon Kim and Alexander M. Rush. 2016. [Sequence-level knowledge distillation](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. [OpenNMT: Open-source toolkit for neural machine translation](#). In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.
- Tuan Le, Marco Bertolini, Frank Noé, and Djork-Arné Clevert. 2021. [Parameterized hypercomplex graph neural networks for graph classification](#). In *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III*, page 204–216, Berlin, Heidelberg. Springer-Verlag.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. [Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning](#).
- 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.
- Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. [STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.
- 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.
- 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.

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.

XMD^{*}: An End-to-End Framework for Interactive Explanation-Based Debugging of NLP Models

Warning: This paper discusses and contains content that may be offensive or upsetting.

Dong-Ho Lee^{1*} Akshen Kadakia^{1*} Brihi Joshi¹ Aaron Chan¹ Ziyi Liu¹ Kiran Narahari¹
Takashi Shibuya² Ryosuke Mitani² Toshiyuki Sekiya² Jay Pujara¹ Xiang Ren¹

¹Department of Computer Science, University of Southern California

²R&D Center Sony Group Corporation

{dongho.lee, akshenhe, brihijos, chanaaro, zliu2803, vnarahar, jpujara, xiangren}@usc.edu

{Takashi.Tak.Shibuya, Ryosuke.Mitani, Toshiyuki.Sekiya}@sony.com

Abstract

NLP models are susceptible to learning spurious biases (*i.e.*, bugs) that work on some datasets but do not properly reflect the underlying task. Explanation-based model debugging aims to resolve spurious biases by showing human users explanations of model behavior, asking users to give feedback on the behavior, then using the feedback to update the model. While existing model debugging methods have shown promise, their prototype-level implementations provide limited practical utility. Thus, we propose XMD^{*}: the first open-source, end-to-end framework for explanation-based model debugging. Given task- or instance-level explanations, users can flexibly provide various forms of feedback via an intuitive, web-based UI. After receiving user feedback, XMD^{*} automatically updates the model in real time, by regularizing the model so that its explanations align with the user feedback. The new model can then be easily deployed into real-world applications via Hugging Face. Using XMD^{*}, we can improve the model’s OOD performance on text classification tasks by up to 18%.¹

1 Introduction

Neural language models have achieved remarkable performance on a wide range of natural language processing (NLP) tasks (Srivastava et al., 2022). However, studies have shown that such NLP models are susceptible to learning spurious biases (*i.e.*, bugs) that work on specific datasets but do not properly reflect the underlying task (Adebayo et al., 2020; Geirhos et al., 2020; Du et al., 2021; Sagawa et al., 2020). For example, in hate speech detection, existing NLP models often associate certain group identifiers (*e.g.*, *black*, *muslims*) with hate speech, regardless of how these words are actually used (Kennedy et al., 2020b) (Fig. 1). This poses se-

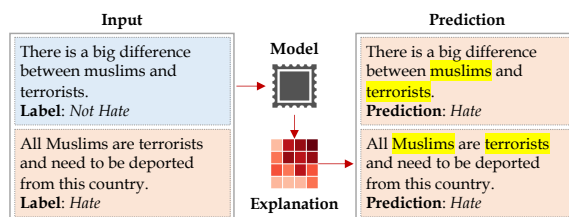


Figure 1: We make predictions on machine-generated examples (Brown et al., 2020; Hartvigsen et al., 2022) using BERT model fine-tuned on HateXplain (Mathew et al., 2021) and show its explanation using integrated gradients (Sundararajan et al., 2017). It shows spurious correlation between a word *muslims* and the label *hate*.

rious concerns about the usage of NLP models for high-stakes decision-making (Bender et al., 2021).

In response, many methods have been proposed for debiasing either the model or the dataset. Model debiasing can be done via techniques like instance reweighting (Schuster et al., 2019), confidence regularization (Utama et al., 2020), and model ensembling (He et al., 2019; Mahabadi and Henderson, 2019; Clark et al., 2019). Dataset debiasing can be done via techniques like data augmentation (Jia and Liang, 2017; Kaushik et al., 2020) and adversarial filtering (Zellers et al., 2018; Le Bras et al., 2020). However, these methods lack knowledge of which spurious biases actually impacted the model’s decisions, which greatly limits their debiasing ability.

On the other hand, *explanation-based model debugging* focuses on addressing spurious biases that actually influenced the given model’s decision-making (Smith-Renner et al., 2020; Lertvitayakumjorn and Toni, 2021; Hartmann and Sonntag, 2022). In this paradigm, a human-in-the-loop (HITL) user is given explanations of the model’s behavior (Sundararajan et al., 2017; Shrikumar et al., 2017) and asked to provide feedback about the behavior. Then, the feedback is used to update the model, in order to correct any spurious biases detected via the user feedback. While existing model debugging methods have shown promise

*Both authors contributed equally.

¹Source code and project demonstration video are made publicly available at <http://inklab.usc.edu/xmd/>

(Idahl et al., 2021; Lertvittayakumjorn et al., 2020; Zylberajch et al., 2021; Ribeiro et al., 2016), their prototype-level implementations provide limited end-to-end utility (*i.e.*, explanation generation, explanation visualization, user feedback collection, model updating, model deployment) for practical use cases.

Given the interactive nature of explanation-based model debugging, it is important to have a user-friendly framework for executing the full debugging pipeline. To achieve this, we propose the EXplanation-Based NLP Model Debugger (**XMD**^{*}). Compared to prior works, **XMD**^{*} makes it simple for users to debug NLP models and gives users significant control over the debugging process (Fig. 2). Given either task (model behavior over all instances) or instance (model behavior w.r.t. a given instance) explanations, users can flexibly provide various forms of feedback (*e.g.*, *add* or *remove* focus on a given token) through an easy-to-use, web-based user interface (UI). To streamline user feedback collection, **XMD**^{*}'s UI presents intuitive visualizations of model explanations as well as the different options for adjusting model behavior (Fig. 3-4). After receiving user feedback, **XMD**^{*} automatically updates the model in real time, by regularizing the model so that its explanations align with the user feedback (Joshi et al., 2022). **XMD**^{*} also provides various algorithms for conducting model regularization. The newly debugged model can then be downloaded and imported into real-world applications via Hugging Face (Wolf et al., 2020). To the best of our knowledge, **XMD**^{*} is the first open-source, end-to-end framework for explanation-based model debugging. We summarize our contributions as follows:

- **End-to-End Model Debugging:** **XMD**^{*} packages the entire model debugging pipeline (*i.e.*, explanation generation, explanation visualization, user feedback collection, model updating, model deployment) as a unified system. **XMD**^{*} is agnostic to the explanation method, user feedback type, or model regularization method. **XMD**^{*} can improve models' out-of-distribution (OOD) performance on text classification tasks (*e.g.*, hate speech detection, sentiment analysis) by up to 18%.
- **Intuitive UI:** **XMD**^{*}'s point-and-click UI makes it easy for non-experts to understand model explanations and give feedback on model behavior.
- **Easy Model Deployment:** Given user feedback, **XMD**^{*} automatically updates the model in real

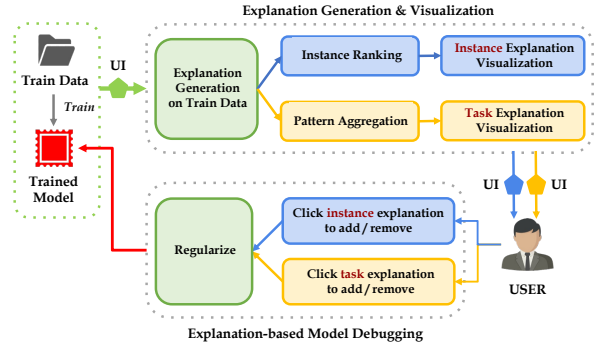


Figure 2: **System Architecture**

time. Users can easily deploy debugged models into real-world applications via Hugging Face.

2 Framework Overview

As shown in Figure 2, our framework consists of three main components: *Explanation Generation*; *Explanation Visualization*; and *Explanation-based Model Debugging*. Here, the explanation generation and debugging process are done on the back-end while visualizing explanations and capturing human feedback on them are done on front-end UI.

Explanation Generation (§3.1) Humans first input the train data and the model trained on the train data into the framework. On the backend, our framework uses a heuristic post-hoc explanation approach on the model to generate rationales for the train data.

Explanation Visualization (§3.2) The framework visualizes the generated rationales through UI in two different ways: an instance explanation, which shows the explanation for each train instance, and a task explanation, which shows words according to their importance to the prediction.

Explanation-Based Model Debugging (§3.3) The human then decides whether to select words for each instance (Instance Explanation) or words that apply to all instances (Task Explanation) to increase or decrease the word importance. When a human clicks a few words to debug and then decides to end the debugging process, the framework retrains the model with a regularization approach and makes the debugged model downloadable.

3 XMD^{*} Framework

In this section, we present each module of **XMD**^{*} in processing order. To start the process, the user needs to place a training dataset \mathcal{D}_T and a classification model \mathcal{M} that is trained on \mathcal{D}_T .

3.1 Explanation Generation

Our explanation generation module outputs rationales from \mathcal{M} . For each instance $\mathbf{x} \in \mathcal{D}_T$, \mathcal{M} generates rationales $\phi(\mathbf{x}) = [\phi(\mathbf{w}_1), \phi(\mathbf{w}_2), \dots, \phi(\mathbf{w}_n)]$ where \mathbf{w}_i denotes the i -th token in the sentence. Each importance score $\phi(\mathbf{w}_i)$ has a score with regard to all the classes. Our module is exploiting $\phi^p(\mathbf{w}_i)$ which the importance score is attributed to model predicted label p . Here, we exploit heuristic methods that assign importance scores ϕ based on gradient changes in \mathcal{M} (Shrikumar et al., 2017; Sundararajan et al., 2017).

3.2 Explanation Visualization

Our framework supports visualizing the generated rationale in two different forms, *instance* and *task* explanations. Instance explanations display word importance scores for model predictions for each train instance, while task explanations aggregate and rank words according to their importance to the predicted label. In this section, we first present a UI for visualizing and capturing human feedback for instance explanations and then a UI for task explanations.

Instance Explanation Figure 3 illustrates how our framework visualizes instance explanations and captures human feedback on them. First, the trained model makes a prediction and generates explanations for one of the train instances that the model correctly predicts: “All muslims are terrorists and need to be deported from this country”. The reason why we present only the instances that the model correctly predicts is that we are asking users to provide feedback for the ground truth label and comparing it with $\phi^p(\mathbf{w}_i)$ which the importance score is attributed to the model predicted label p . If p is not equal to the ground truth label, the human feedback would act as a source of incorrect prediction.

Next, the user is presented with the sentence and its ground truth label on the upper deck (Words Section), and the sentence with highlighted rationales and its predicted label on the lower deck (Model Output Section). Then, the user can choose to select words to decrease or increase its importance toward the ground truth label (Figure 3 (a)). If the user clicks the word (*muslims*) that the model is focusing on to predict *hate*, a small pop-up displaying buttons for operation options (*i.e.*, *add*, *remove* and *reset*) appear. Once the user selects a desired

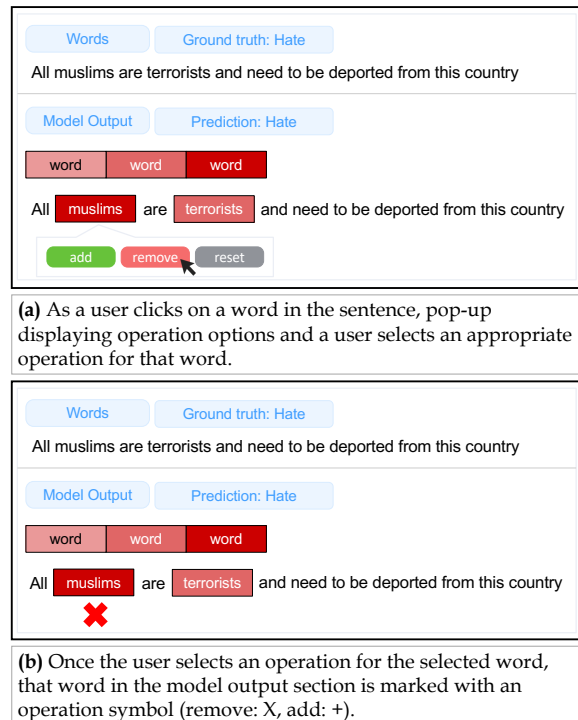


Figure 3: The workflow to provide human feedback on **instance explanations**. Humans provide explanations (*remove* “*muslims*”) for the ground truth label (*hate*).

operation (*remove*) for the selected word (*muslims*) that is not a right reason for *hate*, that word in the model output section is marked with operation symbol ('X' for *remove*, '+' for *add* – Figure 3 (b)). The user may cancel their decision to operation for the word by clicking *reset* in the pop-up.

Task Explanation Figure 4 illustrates how our framework visualizes task explanations and captures human feedback on them. First, task explanations are presented in list format on the left panel in descending order of its importance (Figure 4 (a)). Here, the importance is a score averaged by the word importance score of all examples containing that word. As user clicks on a word in the list, all the examples containing that word are displayed. The user can then choose to two different operations (*remove* and *add*). If user clicks *remove* for the word (*muslims*) that should not be conditioned on any label (both *hate* and *not hate*), the model will consider it as an unimportant word in all cases. Here, we don't need to consider whether the prediction is correct or not since the word is not important for all the cases (Figure 4 (a)). If user clicks *add* for the word that should be useful for the correct prediction, the model will consider it as an important word for the ground truth label. Here, we consider it as an important word only for the

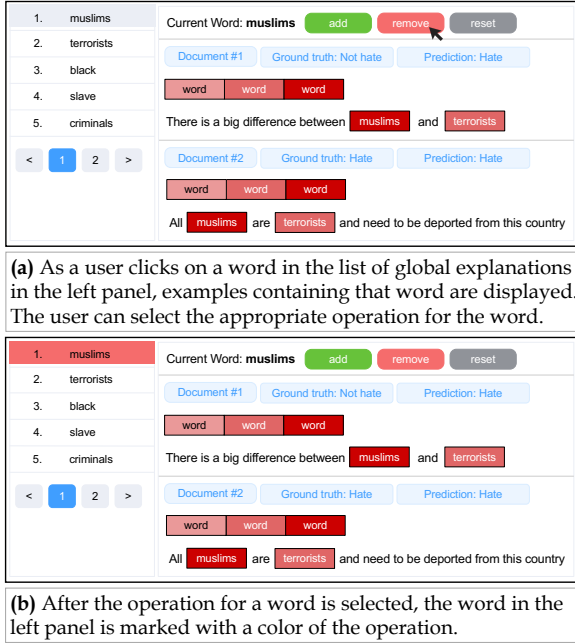


Figure 4: The workflow to provide human feedback on **task explanations**. Humans remove the word (*muslims*) that should not be conditioned on any labels (i.e., *hate*, *not hate*).

correct prediction. After the operation for a word is selected, the word in the left panel is marked with a color of that operation (red for *remove* and green for *add*).

3.3 Explanation-based Model Debugging

Our explanation-based model debugging module is based on explanation regularization (ER) which regularizes model to produce rationales that align to human rationales (Zaidan and Eisner, 2008; Ross et al., 2017; Liu and Avci, 2019a; Ghaeini et al., 2019; Kennedy et al., 2020a; Rieger et al., 2020; Lin et al., 2020; Huang et al., 2021; Joshi et al., 2022). Existing works require the human to annotate rationales for each training instance or apply task-level human priors (e.g., task-specific lexicons) across all training instances before training. Despite its effectiveness, the regularized model may not be fully free of hidden biased patterns. To catch all the hidden biased patterns, our framework asks the human to provide binary feedback (i.e., click to add or remove) given the current model explanations and use them to regularize the model. Here we ask the human to provide feedback to the model in order to output the “**correct prediction**”.

For the instance explanation, as shown in Figure 5, the trained model \mathcal{M} generates rationales $\phi^p(\mathbf{x}) = [\phi^p(w_1), \phi^p(w_2), \dots, \phi^p(w_n)]$, where $\phi^p(w_i)$ denotes the importance score of i -th token

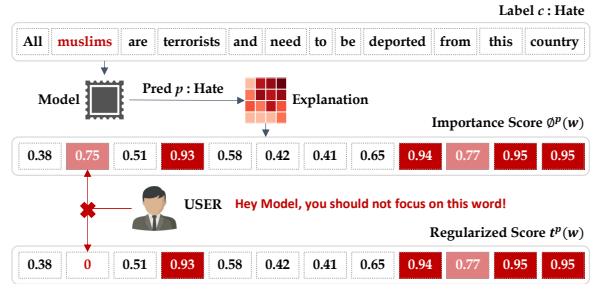


Figure 5: **Instance Explanation-based Model Debugging**. Trained model generates explanations in a form of word importance score $\phi^p(w)$ towards prediction label p . User selects words to *add* or *remove* based on $\phi^p(w)$. The regularization score $t^p(w)$ for the selected words to be removed are 0 while selected words to add are 1.

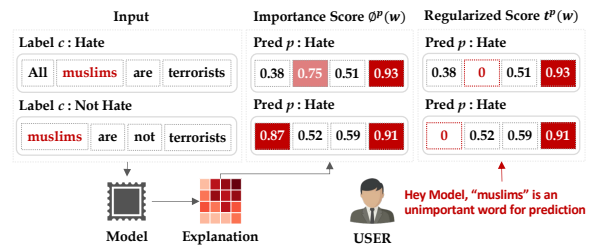


Figure 6: **Task Explanation-based Model Debugging**. Trained model generates explanations in a form of word importance score $\phi^p(w)$ towards prediction label p . As user selects a word to ignore for prediction, the regularization score $t^p(w)$ for the selected word in all the examples that contain that word becomes 0.

in the sentence \mathbf{x} towards model predicted label p . As the user selects a word (*muslims*) that is spuriously correlated with the correct prediction p (*hate*), the regularized score $t^p(w_i)$ where i is a user-selected word index ($w_2 = \text{muslims}$) becomes 0 (See Figure 5). For the task explanation, we aggregate words based on its score averaged by the word importance score of examples containing that word, and present them in a descending order. When the user clicks a word w (*muslims*) to decrease its importance, then regularized score $t^p(w)$ where w is user-selected word ($w = \text{muslims}$) for all the examples become 0 (See Figure 6).

After the click process, the user can start the debugging process based on the examples labeled so far. Here, the learning objective for re-training the model \mathcal{M} is $\mathcal{L} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{ER}}$, where $\mathcal{L}_{\text{task}}$ is a cross-entropy loss for traditional sequence classification tasks and \mathcal{L}_{ER} is an explanation regularization loss which minimizes the distance between $\phi^p(w)$ and $t^p(w_i)$ (Joshi et al., 2022). In this framework, we support two different regularization loss:

Mean Squared Error (MSE) (Liu and Avci, 2019b; Kennedy et al., 2020b; Ross et al., 2017), Mean Absolute Error (MAE) (Rieger et al., 2020).

4 Implementation Details

To start **XMD**^{*}, users should input the trained model following Hugging Face model structure (Wolf et al., 2020). After users input the train data and the model, our framework uses Captum (Kokhlikyan et al., 2020) to generate explanation. For visualizing the explanation and capturing the human feedback, we implement UI using Vue.js². Here, we re-use UI components from LEAN-LIFE, an explanation-based annotation framework (Lee et al., 2020), for capturing human feedback. To train the model with ER, we use PyTorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020).

5 Experiments

We conduct extensive experiments investigating how our debugging process affects the performance on in-distributed (ID) and out-of-distribution (OOD) data, and the model explanation. Here, we present experimental results on sentiment analysis for the instance explanation and hate speech detection for the task explanation. For base model, we use BigBird-Base (Zaheer et al., 2020).

Tasks and Datasets For sentiment analysis, we exploit SST (Socher et al., 2013) as the ID dataset, and Yelp (restaurant reviews) (Zhang et al., 2015), Amazon (product reviews) (McAuley and Leskovec, 2013) and Movies (movie reviews) (Zaidan and Eisner, 2008; DeYoung et al., 2019) as OOD datasets. To simulate human feedback for the instance explanation, we leverage ground truth rationales for SST (Carton et al., 2020) as human feedback. For hate speech detection, we use STF (de Gibert et al., 2018) as the ID dataset, and HatEval (Barbieri et al., 2020), Gab Hate Corpus (GHC) (Kennedy et al., 2018) and Latent Hatred (ElSherief et al., 2021) for OOD datasets. To simulate human feedback for the task explanations, we leverage group identifiers (e.g., *black*, *muslims*) (Kennedy et al., 2020b) as words that need to be discarded for determining whether the instance is hate or not.

ID/OOD Performance Table 1 shows the performance on ID and OOD when regularize on correct

²<https://vuejs.org/>

Regularize	ER Loss	Sentiment Analysis			
		In-distribution		Out-of-Distribution	
		SST	Amazon	Yelp	Movies
None	None	93.4	<u>89.1</u>	89.0	82.0
Correct	MSE	94.7	88.4	<u>91.8</u>	94.5
	MAE	<u>94.0</u>	92.3	94.4	<u>94.0</u>

Table 1: **Instance Explanation** ID/OOD Performance (Accuracy). Best models are bold and second best ones are underlined within each metric.

Regularize	ER Loss	Hate Speech Analysis			
		In-distribution		Out-of-Distribution	
		STF	HatEval	GHC	Latent
None	None	89.5	88.2	64.5	67.2
Correct	MSE	89.2	90.1	62.3	67.9
	MAE	89.1	90.1	59.3	64.9
Incorrect	MSE	88.9	86.3	67.9	70.3
	MAE	89.3	<u>88.8</u>	64.2	67.6
ALL	MSE	90.0	88.4	63.8	67.0
	MAE	<u>89.7</u>	86.9	<u>66.5</u>	<u>70.2</u>

Table 2: **Task Explanation** ID/OOD Performance (Accuracy). Best models are bold and second best ones are underlined within each metric.

predictions using its instance explanation. We see that our framework helps model to not only do much better on ID data, but also generalize well to OOD data. For task explanation, we present performance by regularizing on correct and incorrect prediction and all the instances regardless of prediction. Table 2 presents the performance with *remove* operations for task explanations (i.e., group identifiers) for incorrect predictions, correct predictions, and for all instances, respectively. We observe that our framework helps model not to focus on the words that should not be conditioned on any label and lead to performance enhancement on both ID and OOD data.

Efficiency To quantify the advantage that **XMD**^{*} provides, we compare the time taken to annotate instances using **XMD**^{*} versus traditional labelling for instance explanations. While **XMD**^{*} requires humans to interact with a trained model and decrease or increase importance scores of words, traditional labelling is not model-in-the-loop in nature, and requires users to directly annotate binary importance scores to words in the instance (DeYoung et al., 2019; Carton et al., 2020). We ask two graduate students to annotate 50 instances, using the traditional and the **XMD**^{*} labelling methods. For both of these labelling settings, we ensure that there is no overlap between the instances, so as to avoid familiarity and record

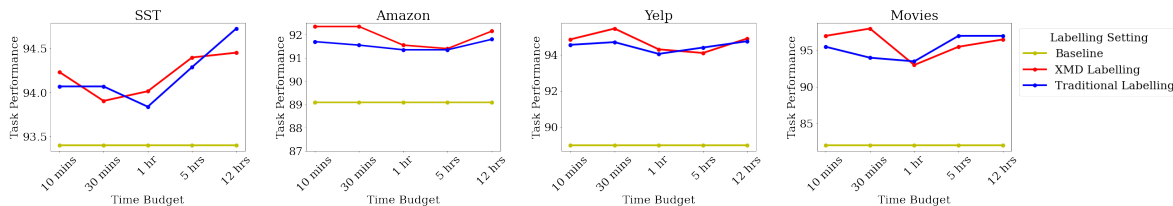


Figure 7: **Time Efficiency:** This simulation assumes two annotators annotating instances in parallel with a strict time budget, using the traditional labelling (~ 110 s/instance) or **XMD*** labelling (~ 60 s/instance) methods.

the time taken to annotate each instances. Upon aggregating across all instances and both annotators, it is found that one instance takes ~ 60 seconds and ~ 110 seconds to be annotated using the framework and the traditional labelling method respectively. Using this time estimate, we simulate the time-efficiency of these two labelling methods with varying amounts of time budgets for annotations. Figure 7 presents our results for this experiment. We note that although both labelling methods outperforms the baseline of no explanation annotation, using **XMD*** is particularly helpful when the time budget given is limited (< 1 hour), especially in the OOD setting (Amazon, Yelp, Movies datasets).

6 Related Work

Spurious Bias Mitigation Recent studies have explored mitigating spurious biases in NLP models. One of the research lines is a dataset debiasing such as adversarial filtering (Zellers et al., 2018; Le Bras et al., 2020) or data augmentation using adversarial data (Jia and Liang, 2017) and counterfactual data (Kaushik et al., 2020). However, creating such datapoints are challenging since they require an exhaustive understanding of the preconceived notions that may cause such spurious biases and the collecting cost is expensive. Another line of research is robust learning techniques such as instance reweighting (Schuster et al., 2019), confidence regularization (Utama et al., 2020), and model ensembling (He et al., 2019; Mahabadi and Henderson, 2019; Clark et al., 2019).

Explanation-Based Model Debugging Many works have explored explanation-based debugging of NLP models, mainly differing in how model behavior is explained, how HITL feedback is provided, and how the model is updated (Lertvittayakumjorn and Toni, 2021; Hartmann and Sonntag, 2022; Balkir et al., 2022). Model behavior can be explained using instance (Idahl et al., 2021; Koh and Liang, 2017; Ribeiro et al., 2016) or task (Lertvittayakumjorn et al., 2020; Ribeiro

et al., 2018) explanations, typically via feature importance scores. HITL feedback can be provided by modifying the explanation’s feature importance scores (Kulesza et al., 2009, 2015; Zylberajch et al., 2021) or deciding the relevance of high-scoring features (Lu et al., 2022; Kulesza et al., 2010; Ribeiro et al., 2016; Teso and Kersting, 2019). The model can be updated by directly adjusting the model parameters (Kulesza et al., 2009, 2015; Smith-Renner et al., 2020), improving the training data (Koh and Liang, 2017; Ribeiro et al., 2016; Teso and Kersting, 2019), or influencing the training process (Yao et al., 2021; Cho et al., 2019; Stumpf et al., 2009). In particular, explanation regularization (ER) influences the training process so that the model’s explanations align with human explanations (Joshi et al., 2022; Ross et al., 2017; Kennedy et al., 2020a; Rieger et al., 2020; Liu and Avci, 2019a; Chan et al., 2022).

Our **XMD*** system is agnostic to the choice of explanation method or HITL feedback type, while updating the model via ER. Compared to prior works, **XMD*** gives users more control over the interactive model debugging process. Given either global or local explanations, users can flexibly provide various forms of feedback via an intuitive, web-based UI. After receiving user feedback, **XMD*** automatically updates the model in real time. The debugged model can then be downloaded and imported into real-world applications via Hugging Face (Wolf et al., 2020).

7 Conclusion

In this paper, we propose an open-source and web-based explanation-based NLP Model Debugging framework **XMD*** that allows user to provide various forms of feedback on model explanation. This debugging process guides the model to make predictions with the correct reason and lead to significant improvement on model generalizability. We hope that **XMD*** will make it easier for researchers and practitioners to catch spurious correlations in the model and debug them efficiently.

References

- Julius Adebayo, Michael Muelly, Ilaria Lippardi, and Been Kim. 2020. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429*.
- Esma Balkir, Svetlana Kiritchenko, Isar Nejadgholi, and Kathleen C Fraser. 2022. Challenges in applying explainability methods to improve the fairness of nlp models. *arXiv preprint arXiv:2206.03945*.
- Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. **TweetEval: Unified benchmark and comparative evaluation for tweet classification**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.
- 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.
- 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.
- Samuel Carton, Anirudh Rathore, and Chenhao Tan. 2020. Evaluating and characterizing human rationales. *arXiv preprint arXiv:2010.04736*.
- Aaron Chan, Maziar Sanjabi, Lambert Mathias, Liang Tan, Shaoliang Nie, Xiaochang Peng, Xiang Ren, and Hamed Firooz. 2022. Unirex: A unified learning framework for language model rationale extraction. *International Conference on Machine Learning*.
- Minseok Cho, Gyeongbok Lee, and Seung-won Hwang. 2019. Explanatory and actionable debugging for machine learning: A tableqa demonstration. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1333–1336.
- Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don’t take the easy way out: Ensemble based methods for avoiding known dataset biases. 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 4069–4082, Hong Kong, China. Association for Computational Linguistics.
- Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. **Hate speech dataset from a white supremacy forum**. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20, Brussels, Belgium. Association for Computational Linguistics.
- Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2019. Eraser: A benchmark to evaluate rationalized nlp models. *arXiv preprint arXiv:1911.03429*.
- Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. 2021. **Towards interpreting and mitigating shortcut learning behavior of NLU models**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929, Online. Association for Computational Linguistics.
- 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.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.
- Reza Ghaeini, Xiaoli Z Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Saliency learning: Teaching the model where to pay attention. *arXiv preprint arXiv:1902.08649*.
- Mareike Hartmann and Daniel Sonntag. 2022. A survey on improving nlp models with human explanations. *arXiv preprint arXiv:2204.08892*.
- Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. **ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.
- He He, Sheng Zha, and Haohan Wang. 2019. **Unlearn dataset bias in natural language inference by fitting the residual**. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 132–142, Hong Kong, China. Association for Computational Linguistics.
- Quzhe Huang, Shengqi Zhu, Yansong Feng, and Dongyan Zhao. 2021. Exploring distantly-labeled rationales in neural network models. *arXiv preprint arXiv:2106.01809*.
- Maximilian Idahl, Lijun Lyu, Ujwal Gadiraju, and Avishek Anand. 2021. Towards benchmarking the utility of explanations for model debugging. *arXiv preprint arXiv:2105.04505*.

- Robin Jia and Percy Liang. 2017. [Adversarial examples for evaluating reading comprehension systems](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.
- Brihi Joshi, Aaron Chan, Ziyi Liu, Shaoliang Nie, Maziar Sanjabi, Hamed Firooz, and Xiang Ren. 2022. Er-test: Evaluating explanation regularization methods for nlp models. *arXiv preprint arXiv:2205.12542*.
- Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. [Learning the difference that makes a difference with counterfactually-augmented data](#). In *International Conference on Learning Representations*.
- Brendan Kennedy, Mohammad Atari, Aida M Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs, Shreya Havaldar, Gwennyth Portillo-Wightman, Elaine Gonzalez, and et al. 2018. [Introducing the gab hate corpus: Defining and applying hate-based rhetoric to social media posts at scale](#).
- Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. 2020a. Contextualizing hate speech classifiers with post-hoc explanation. *arXiv preprint arXiv:2005.02439*.
- Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. 2020b. [Contextualizing hate speech classifiers with post-hoc explanation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5435–5442, Online. Association for Computational Linguistics.
- Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. [Captum: A unified and generic model interpretability library for pytorch](#).
- Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.
- Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. 2010. Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 41–48. IEEE.
- Todd Kulesza, Weng-Keen Wong, Simone Stumpf, Stephen Perona, Rachel White, Margaret M Burnett, Ian Oberst, and Amy J Ko. 2009. Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 187–196.
- Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR.
- Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020. [LEAN-LIFE: A label-efficient annotation framework towards learning from explanation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 372–379, Online. Association for Computational Linguistics.
- Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. [FIND: Human-in-the-Loop Debugging Deep Text Classifiers](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online. Association for Computational Linguistics.
- Piyawat Lertvittayakumjorn and Francesca Toni. 2021. Explanation-based human debugging of nlp models: A survey. *Transactions of the Association for Computational Linguistics*, 9:1508–1528.
- Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. 2020. Triggerer: Learning with entity triggers as explanations for named entity recognition. *arXiv preprint arXiv:2004.07493*.
- Frederick Liu and Besim Avci. 2019a. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*.
- Frederick Liu and Besim Avci. 2019b. [Incorporating priors with feature attribution on text classification](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy. Association for Computational Linguistics.
- Jinghui Lu, Linyi Yang, Brian Mac Namee, and Yue Zhang. 2022. A rationale-centric framework for human-in-the-loop machine learning. *arXiv preprint arXiv:2203.12918*.
- Rabeeh Karimi Mahabadi and James Henderson. 2019. Simple but effective techniques to reduce biases.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14867–14875.

- Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. 2020. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International conference on machine learning*, pages 8116–8126. PMLR.
- Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.
- Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8346–8356. PMLR.
- Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. 2019. [Towards debiasing fact verification models](#). 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 3419–3425, Hong Kong, China. Association for Computational Linguistics.
- Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.
- Alison Smith-Renner, Ron Fan, Melissa Birchfield, Tongshuang Wu, Jordan Boyd-Graber, Daniel S Weld, and Leah Findlater. 2020. No explainability without accountability: An empirical study of explanations and feedback in interactive ml. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.
- Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International journal of human-computer studies*, 67(8):639–662.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.
- Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245.
- Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Mind the trade-off: Debiasing nlu models without degrading the in-distribution performance. *arXiv preprint arXiv:2005.00315*.
- 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.
- Huihan Yao, Ying Chen, Qinyuan Ye, Xisen Jin, and Xiang Ren. 2021. Refining language models with compositional explanations. *Advances in Neural Information Processing Systems*, 34:8954–8967.

- 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. In *NeurIPS*.
- Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 conference on Empirical methods in natural language processing*, pages 31–40.
- Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. [SWAG: A large-scale adversarial dataset for grounded commonsense inference](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. [Character-level Convolutional Networks for Text Classification](#). *arXiv:1509.01626 [cs]*.
- Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. [HILDIF: Interactive debugging of NLI models using influence functions](#). In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 1–6, Online. Association for Computational Linguistics.

OpenDelta: A Plug-and-play Library for Parameter-efficient Adaptation of Pre-trained Models

Shengding Hu^{1,2}, Ning Ding^{1,2}, Weilin Zhao^{1,2}, Xingtai Lv¹, Zhen Zhang¹
Zhiyuan Liu^{1,2,3*}, Maosong Sun^{1,2}

¹Dept. of Comp. Sci. & Tech., IAI, BNRIST, Tsinghua University, Beijing

²International Innovation Center of Tsinghua University, Shanghai, China

³Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University

hsd20@mails.tsinghua.edu.cn

Abstract

The scale of large pre-trained models (PTMs) poses significant challenges in adapting to downstream tasks due to the high optimization overhead and storage costs associated with full-parameter fine-tuning. To address this, many studies explore parameter-efficient tuning methods, also framed as “delta tuning” in Ding et al. (2022), which updates only a small subset of parameters, known as “delta modules”, while keeping the backbone model’s parameters fixed. However, the practicality and flexibility of delta tuning have been limited due to existing implementations that directly modify the code of the backbone PTMs and hard-code specific delta tuning methods for each PTM. In this paper, we present OpenDelta¹, an open-source library that overcomes these limitations by providing a plug-and-play implementation of various delta tuning methods. Our novel techniques eliminate the need to modify the backbone PTMs’ code, making OpenDelta compatible with different, even novel PTMs. OpenDelta is designed to be simple, modular, and extensible, providing a comprehensive platform for researchers and practitioners to adapt large PTMs efficiently.

1 Introduction

With the rapid development of self-supervised learning methods in the realm of deep learning, especially pre-training techniques (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018), foundational pre-trained models (Bommasani et al., 2021) (PTMs) have become a common cornerstone for numerous downstream tasks. And as a result, research into large-scale PTMs has flourished.

Nevertheless, the ever-expanding scale of PTMs also poses substantial obstacles in practical use. In traditional model adaptation, all the parameters

of the PTMs are optimized for each downstream task, which becomes increasingly impractical as the model scales. Firstly, optimizing all the parameters incurs prohibitive computing and memory consumption; secondly, storing a fine-tuned model instance for each task or experiment significantly amplifies the storage cost.

To address these challenges, researchers have developed parameter-efficient methods for model adaptation. Such methods keep the parameters of the main model fixed and update only a small subset of parameters during adaptation. This approach, known as “delta tuning”, is described and surveyed in Ding et al. (2022). Different delta tuning methods have been proposed, with varying types and positions of “delta modules”. For example, Adapter module (Houlsby et al., 2019) is composed of two low-dimensional linear projection layers with an activation function, while LoRA (Hu et al., 2021) module introduces a low-rank decomposition for the weight matrix. BitFit (Zaken et al., 2021), on the other hand, specifies the bias vector in PTMs as the delta modules. The delta module can be applied to different positions (Rücklé et al., 2020; He et al., 2022; Hu et al., 2022) to achieve either better performance or efficiency.

Theoretically, incorporating most delta tuning methods would necessitate restructuring the backbone model, a requirement conventionally achieved through direct code manipulation. While this method may seem simple, it carries several disadvantages. Primarily, it lacks flexibility, as delta modules can theoretically be implemented in various positions, making modifications to each position in the backbone model code a cumbersome task. Additionally, this method is not scalable, as accommodating delta tuning for newly introduced PTMs requires fresh code modifications, posing a challenge for researchers and engineers.

In this paper, we present a novel approach to implement delta tuning methods. Our approach

* corresponding author liuzy@tsinghua.edu.cn

¹GitHub Repo <https://github.com/thunlp/OpenDelta>, Demo Video <https://rb.gy/qjvpav>.

modifies the backbone model’s architecture after it is loaded into the memory. We propose four essential techniques, namely named-based addressing, dynamic tensor re-routing, runtime initialization, and a visualization system. Using these key techniques, we build OpenDelta, an open-source toolkit for delta tuning without modifying the backbone model code. OpenDelta has several key features. Firstly, it is simple to use. Migrating from existing full-parameter training to delta tuning requires as few as three lines of code. For beginners or engineers, we also support automatic delta model construction. Secondly, it is modular, with delta modules implemented as independent sub-modules that can be attached to or detached from the backbone models. This feature allows different delta modules to coexist and cooperate in the same backbone model and serves multiple tasks flexibly. Thirdly, OpenDelta is highly extensible, supporting pre-trained models in a wide range of frameworks, including both official implementations from the Huggingface Library (Wolf et al., 2019) and customized PTMs. It can potentially be used with newly emerged PTMs and integrated with other PTMs’ frameworks for efficient training, such as the parallel training framework.

2 Related Work

Our work is related to delta tuning, more specifically, the implementation of delta tuning methods.

Delta Tuning. Delta tuning refers to the parameter-efficient method for tuning a large PTM. Different delta tuning methods (Houlsby et al., 2019; Zaken et al., 2021; Li and Liang, 2021; Hu et al., 2021; Mahabadi et al., 2021; Sung et al., 2022) differ in both the architecture of the delta module and the positions that the delta modules are integrated into the backbone model. Various works have attempted to connect these disparate delta tuning approaches under a unified perspective (He et al., 2022; Ding et al., 2022; Hu et al., 2022). In our work, we draw inspiration from this unified viewpoint and aim to devise a framework that can support different delta tuning methods within the same pipeline. Our library includes the most popular delta tuning methods and is amenable to new methods as they emerge.

Implementation of Delta tuning. Previous implementation frameworks for delta tuning relied on the code modification approach. For example, AdapterHub (Pfeiffer et al., 2020) copies a specific

version of Huggingface transformers Library (Wolf et al., 2019) and implement several popular delta tuning methods for a set of pre-defined PTMs. LoRA (Hu et al., 2021) implements a limited library of LoRA linear layers. These methods are model-specific and involve hard-coded implementations, which restrict their usability across various PTMs. In contrast, OpenDelta represents a significant advancement as it requires no code changes to the backbone model, making it highly versatile and broadly applicable.

3 Motivation

In this section, we begin by presenting the unified formulation of delta tuning. Then we underscore a set of crucial characteristics of delta tuning, focusing on the implementation aspect, which emphasizes the pressing need for a novel toolkit to aid in the research and advancement of delta tuning approaches.

3.1 Unified Formulation of Delta Tuning

Although delta tuning is principally not limited to a specific type of neural networks, currently almost all the delta tuning methods are applied to PTMs (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019; Brown et al., 2020) with the Transformers architecture (Vaswani et al., 2017). A PTM \mathcal{M} parameterized by Θ is composed of multiple sub-modules m , where the hidden representations \mathbf{h} are passed through the sub-module to produce new hidden representation \mathbf{h}' , i.e., $\mathbf{h}' = m(\mathbf{h})$.

The adaptation of a PTM \mathcal{M} to downstream tasks is to update the original parameters Θ into Θ' . In full-parameter fine-tuning, all parameters can be updated, i.e., potentially, $|\Delta\Theta| = |\Theta|$. In contrast, delta tuning only updates a small fraction of parameters, i.e., $|\Delta\Theta| \ll |\Theta|$.

Despite the drastic difference in the specific form of the delta tuning methods, He et al. (2022) unify them into special forms of modifications $\Delta\mathbf{h}$ to the hidden representation \mathbf{h} . The $\Delta\mathbf{h}$ is generated by passing a hidden state \mathbf{h}_δ to a *delta module* m_δ . Formally,

$$\mathbf{h} \leftarrow \mathbf{h} + \Delta\mathbf{h} = \mathbf{h} + m_\delta(\mathbf{h}_\delta), \quad (1)$$

where \leftarrow denotes a replacement of the original \mathbf{h} , and \mathbf{h}_δ can be the same as or different to \mathbf{h} .

3.2 Key Features for Delta Tuning

Several key features of delta tuning methods can be observed from Eq.(1).

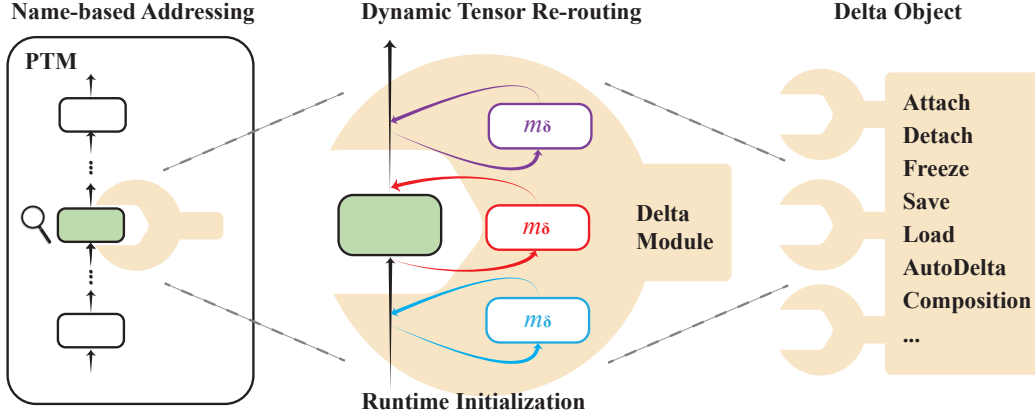


Figure 1: The overall framework of OpenDelta. The construction of delta object happens after the backbone model is loaded.

Tensor Re-routing. The first feature of delta tuning is the ability to redirect the flow of hidden states. In a pre-trained model, the flow of hidden states forms a static graph, with the hidden states serving as nodes and sub-modules acting as transformations on the edges. As shown in Eq.(1), the introduction of the edge transformation m_δ redirects node h_δ and injects it into another node h , creating a new flow of hidden states that is not present in the original model architecture. The implementation of OpenDelta should achieve such tensor re-routing without hard-coding them.

Flexibility. Eq.(1) allows for the input hidden states and output hidden states to be located at any position in the backbone model \mathcal{M} . For example, AdapterDrop (Rücklé et al., 2021) observes that only applying delta modules to the upper half of Transformer layers yields better results than the lower half. The flexibility of applied positions provides remarkable opportunities to explore the potential structure of delta modules (Hu et al., 2022). However, it also presents a challenge for the implementation to be able to achieve flexibility in practice that matches the theoretical framework.

Compositionality. Different delta tuning methods can co-exist or even be combined in the same backbone model (Hu et al., 2022), potentially boosting performance or supporting multitask learning (Pfeiffer et al., 2021). Thus, it is crucial to enable easy and independent implementation of each delta tuning method, while also allowing for the flexible composition of multiple modules.

Dynamism. It is common for the backbone PTM to serve as a central model for multiple tasks in delta tuning. To serve a specific task, delta modules are attached to the backbone model, creating a

task-specific expert. When the delta modules are detached, the backbone models revert back to their original function as general language models. This dynamic nature of delta tuning-based task adaptation should be incorporated into OpenDelta.

4 OpenDelta

In light of the aforementioned key features of delta tuning, we present OpenDelta. We will begin by presenting an overview of OpenDelta. Following that, we will delve into the key implementations of this framework.

4.1 Framework

To perform delta tuning, two prerequisites are required: a pre-trained language model \mathcal{M} and the “modified modules”, which are a user-specified list of sub-modules m_i to which the delta modules should be applied. Our target is to construct a *delta object*. Our objective is to create a delta object, which is a collection of delta modules typically located at various positions within \mathcal{M} and serves as a whole to adapt the PTM to downstream tasks. We follow three steps to create a delta object. Firstly, we use *name-based addressing* to obtain the pointers to the modified modules. Secondly, we construct a delta object comprising uninitialized delta modules. Thirdly, we modify the route of tensors in the modified modules into the delta modules using a *dynamic tensor re-routing* technique. After the updated route of the hidden state is established, we perform *runtime initialization* to initialize the delta object.

After the delta object is constructed, we attach it to the backbone model. Then, we provide a simple functional interface to turn off the gradient

Method	Formulation	Default Positions	Route	Runtime Initialization
LoRA	$m_\delta(\mathbf{h}_{in}) = \mathbf{h}_{in}\mathbf{A}\mathbf{B}$	Query, Value	Eq.(4)	N
Adapter	$m_\delta(\mathbf{h}_{out}) = \sigma(\mathbf{h}_{out}\mathbf{W}_1)\mathbf{W}_2$	ATTN, FFN	Eq.(3)	Y
Bitfit	$m_\delta(\mathbf{h}_{out}) = \mathbf{b}$	ATTN, FFN, LayerNorm	Eq.(3)	N
Prefix Tuning	$m_\delta(\mathbf{h}_{out}) = [\text{MLP}(\mathbf{p}); \mathbf{h}_{out}]$	Key, Value	Eq.(3)	Y

Table 1: Delta tuning methods and their characteristics. Default positions refer to the positions that the delta modules are attached to when no specific sub-modules are designated. $\mathbf{A}, \mathbf{B}, \mathbf{W}_1, \mathbf{W}_2$ are weight matrices, \mathbf{b} is the bias vector. $\text{MLP}(\cdot)$ is a multi-layer perception network. $[\cdot; \cdot]$ denotes the concatenation of tensors. σ is the activation function. Runtime Initialization shows whether the implementation uses this technique in OpenDelta.

computation in the backbone models and only compute the gradient of parameters in the delta object. After the training is complete, we provide a simple interface for saving only the delta objects, which significantly reduces the storage requirements for the backbone model.

The overall framework of OpenDelta is shown in Figure 1. Next, we introduce the key implementations that support the construction of delta objects.

4.2 Key Implementations

The above framework is achieved by four key implementations, i.e., name-based addressing, dynamic tensor re-routing, runtime initialization, and visualization system.

Name-based Addressing. Firstly, we need to obtain a pointer to the desired sub-modules which are applied with the delta modules. In practice, we can effectively retrieve the pointer by using the name of the sub-module. Since the sub-modules are organized in a tree structure, we perform a depth-first search to find the sub-modules that match the provided name. This search results in a full path consisting of all the names from the root to the matched sub-module, accurately matching the sub-module. However, directly writing the full path to the sub-modules can be impractical, so we design several simplifications to make addressing easier and more human-readable². One such simplification involves taking advantage of the repetitiveness of transformer layers, which many delta tuning methods address by adding delta modules to the same type of sub-modules in each layer. For example, when users specify `attention`, they likely intend to apply delta modules to the attention sub-modules in all transformer layers. To address this need, we provide a tail-matching mechanism that automatically matches the sub-modules based on their names. For more complex configurations

²<https://opendelta.readthedocs.io/en/latest/notes/namebasedaddr.html>

of positions, we allow matching based on regular expressions and web-based selection using our custom-designed web interface.

Dynamic Tensor Re-routing. A fundamental distinction that sets OpenDelta apart from other implementations is its ability to add delta modules without requiring any modifications to the code of the backbone modules. This feature necessitates a dynamic rerouting of tensors through the delta modules and back into the backbone model. To achieve this rerouting, we wrap the original forward function of a sub-module with a wrapper function and replace the original forward function with the wrapper function. To ensure seamless replacement, we utilize a decorator to inherit the original function’s attributes, including the I/O, doc string, etc. Within the wrapped function, we implement three distinct routes of the hidden states, taking into account the order of the original sub-module and the delta module. The first route utilizes the input hidden state \mathbf{h}_{in} of m_i as both the modification target and the input to the delta module. We pass it through the delta module to get the output $m_\delta(\mathbf{h}_{in})$, and merge it to \mathbf{h}_{in} . Formally,

$$\mathbf{h}_{in} \leftarrow \mathbf{h}_{in} + m_\delta(\mathbf{h}_{in}). \quad (2)$$

The second route employs the output hidden state \mathbf{h}_{out} of m_i as the modification target:

$$\mathbf{h}_{out} \leftarrow \mathbf{h}_{out} + m_\delta(\mathbf{h}_{out}). \quad (3)$$

The third route leverages the input hidden state \mathbf{h}_{in} as the input to the delta module, and sets the output hidden state \mathbf{h}_{out} as the modification target:

$$\mathbf{h}_{out} \leftarrow \mathbf{h}_{out} + m_\delta(\mathbf{h}_{in}). \quad (4)$$

While these three routes do not necessarily encompass all possible relationships between the delta module and the backbone model, they are sufficient to support most popular delta tuning methods (as illustrated in Table 1). However, we

```

1 model = AutoModel.from_pretrained("bert-base-cased")
2
3 + from bigmodelvis import Visualization
4 + Visualization(model).structure_graph()
5 + from opendelta import LoraModel
6 + delta_model = LoraModel(backbone_model=model, modified_modules=["output.dense"
7 + , "query"])
8 + delta_model.freeze_module(exclude=["deltas", "pooler"], set_state_dict=True)
9 + Visualization(model).structure_graph()
10 trainer.train()

```

Figure 2: An example of basic usage of OpenDelta. ‘+’ sign indicates the additional code needed to enable delta tuning. Note that the visualization can be optional if you are familiar with the backbone model.

remain open to the possibility of incorporating additional routes as needed.

Runtime Initialization. To ensure that weight matrices in the delta module match the hidden states in terms of shape and dimension, we must account for hidden states whose shapes are not specified in the model configuration. In traditional implementations, this requires manually examining the code of the backbone model. However, OpenDelta automates this process by passing a pseudo input through the backbone model, allowing the shapes of the hidden states to be automatically determined as they propagate from the input to the output.

Visualization System. As delta tuning provides flexibility and dynamism, it is essential to ensure the correct construction of delta objects by verifying that delta modules are added as specified. However, direct printing of large pre-trained models results in massive outputs. To address this, we provide a visualization system that leverages repetition in transformer architecture. Specifically, we collapse the repetitive layers and neatly print the parameters’ information. With the addition of delta modules to the backbone model, users can easily observe the changes made in the model through visualization. An example of visualization can be seen in Figure 3. As the visualization system is useful beyond delta tuning, it has been separated into an independent package named “bigmodelvis”³.

a

5 Usage

In this section, we provide the use cases of OpenDelta which demonstrate the three characteristics of OpenDelta, i.e., simplicity, modularity, and extensibility.

³<https://pypi.org/project/bigmodelvis/>

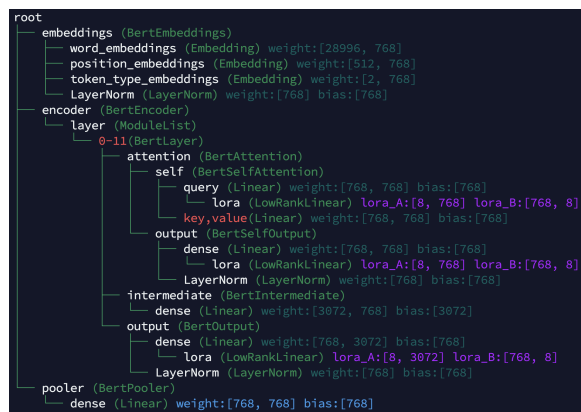


Figure 3: The visualization of the backbone model’s status after the LoRA modules are attached.

5.1 Simplicity

Migrating from Fine-tuning. To facilitate the migration from existing full-parameter fine-tuning to delta tuning, only a few lines of code modifications are required, as exemplified in Figure 2. Initially, in the traditional full-parameter fine-tuning, the PTM is loaded from external libraries, such as Hugging-face Transformers (Line 1), and train the model (Line 10). To introduce delta tuning, line 3-8 are added and executed. To begin with, an optional step is to visualize the backbone model to identify the target “modified_modules”. Then, a delta object, such as LoRA, is created and attached to the backbone model. Subsequently, the model parameters, excluding the delta modules and the randomly initialized classification head, are frozen. The “set_state_dict=True” parameter is employed to remove the non-trainable parameters from the model checkpoint. Lastly, the sub-modules of the backbone are visualized to verify the successful creation and attachment of the delta modules. An example of the visualization results is depicted in Figure 3.

AutoDelta Mechanism. The implementation of OpenDelta supports highly intricate designs of

```

1 def multi_task(delta_model, input_text):
2     global model # We use the same backbone model across tasks.
3     delta_model.attach()
4     print(tokenizer.decode(model.generate(input_ids=tokenize(input_text))))
5     delta_model.detach()
6 multi_task("What the common career of Newton ad einstein?", spelling_delta)
7 # >>> "What was the common career of Newton and Einstein?"
8 multi_task("What was the common career of Newton and Einstein?", topic_delta)
9 # >>> "The question's topic is science."
10 multi_task("What was the common career of Newton and Einstein?", question_delta
11 )
12 # >>> "Physicists."

```

Figure 4: Multitask learning via OpenDelta. Due to space limitations, we retain only the core code. For detailed code, please refer to the OpenDelta documentation. Strings after “>>>” demonstrate the output of the model.

delta modules, catering to diverse experimental requirements. Nonetheless, it is desirable to provide a default configuration of delta modules for practitioners who may not be well-versed in the mechanism of delta tuning. However, the naming conventions of sub-modules differ significantly among various backbone models, despite their shared transformer architecture. To tackle this issue, we establish a common name convention and employ a mapping technique to map the model-specific name convention to the common one⁴. This enables the AutoDelta mechanism to be supported seamlessly. Figure 5 exemplifies that, once the type of the delta tuning method is specified, the delta modules will be attached to the backbone model in default positions and with appropriate hyper-parameters. We have listed the default configurations of each delta tuning method in Table 1. Furthermore, the AutoDelta mechanism facilitates the loading of fine-tuned checkpoints of delta modules, without explicit knowledge of the type and hyper-parameters of the delta modules.

```

1 from opendelta import AutoDeltaModel,
2   AutoDeltaConfig
3 # construct a new delta using the
4 # default configuration.
5 delta_config = AutoDeltaConfig.
6   from_dict({"delta_type": "lora"})
7 delta_model = AutoDeltaModel.
8   from_config(delta_config,
9   backbone_model)
10 # load the delta checkpoint.
11 delta = AutoDeltaModel.from_finetuned(
12   "save_dir", backbone_model)

```

Figure 5: An example of using AutoDelta mechanism.

5.2 Modularity

The second notable attribute of OpenDelta is modularity. It affords the capacity to independently

attach and detach each delta object from the backbone model, thereby providing the possibility of multi-task serving with a single backbone model. Specifically, suppose data pertaining to various tasks are presented sequentially, wherein each data triggers the attachment of a corresponding delta object to the backbone model for processing, and once completed, the delta object is detached. A case that illustrates this functionality is illustrated in Figure 4, where three tasks are process sequentially using a single backbone model.

5.3 Extensibility

Delta tuning is one of the important techniques that enables the use of large PTMs, and as such, we make efforts to ensure its compatibility with other techniques such as model acceleration and multi-GPU training. Specifically, we currently provide support for the BMTrain framework⁵ with ZeRO-3 optimization enabled (Rajbhandari et al., 2020). It is also worth noting that we plan to expand our support for additional model-acceleration frameworks in the future.

6 Conclusion

In summary, OpenDelta is a plug-and-play library for delta tuning, offering an intuitive and modular solution to adapt large PTMs using delta tuning without the need for code modifications. The library’s user-friendliness, flexibility, and extensibility make it accessible and useful for both researchers and engineers. In the future, we plan to continuously update the library with new delta tuning methods and ensure its compatibility with the latest versions of other major PTMs libraries.

⁴<https://opendelta.readthedocs.io/en/latest/notes/unifiname.html>

⁵<https://github.com/OpenBMB/BMTrain>

7 Acknowledgements

This work is supported by the National Key RD Program of China (No.2022ZD0116312), National Natural Science Foundation of China (No. 62236004), Major Project of the National Social Science Foundation of China (No. 22ZD298).

Limitations

Although we believe that OpenDelta is simple, easy to use, flexible, and extensible since it does not require code modification, it is still limited by many implementation details. For example, some delta tuning methods, such as Prefix Tuning, are limited by theory and can only be used in Attention layers, making them unable to be arbitrarily specified. This is also why we did not use it as an example in this paper. On the other hand, some base models differ significantly from mainstream implementations, making it difficult to use the AutoDelta mechanism. Therefore, we maintain a list of tested models that can use AutoDelta, while other models may still use OpenDelta in a customized manner. Thirdly, while theoretically compatible with acceleration frameworks other than BMTrain, such as DeepSpeed, there are some implementation details that currently limit the compatibility of some functions. We will do our best to communicate with the maintainer of those packages to increase compatibility.

Ethical Consideration

In the writing process of this paper, ChatGPT (OpenAI, 2022) was utilized for revision and refinement. However, the authors can guarantee that each sentence in this paper has been thoroughly reviewed and checked to accurately convey the authors' intended meaning.

References

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the 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.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

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*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for NLP](#). In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *ArXiv preprint, abs/2106.09685*.

Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. 2022. Sparse structure search for delta tuning. In *In proceedings of NeurIPS*.

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.
- 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*, abs/1907.11692.
- Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. [Compacter: Efficient low-rank hypercomplex adapter layers](#). *ArXiv preprint*, abs/2106.04647.
- OpenAI. 2022. [Chatgpt: Optimizing language models for dialogue](#).
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics*.
- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. [AdapterFusion: Non-destructive task composition for transfer learning](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.
- 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.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *ArXiv preprint*, abs/1910.10683.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.
- Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. [Adapterdrop: On the efficiency of adapters in transformers](#). *arXiv preprint arXiv:2010.11918*.
- Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. [AdapterDrop: On the efficiency of adapters in transformers](#). In *Proceedings of EMNLP*, pages 7930–7946.
- Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. [Lst: Ladder side-tuning for parameter and memory efficient transfer learning](#). *arXiv preprint arXiv:2206.06522*.
- 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, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. [Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models](#). *ArXiv preprint*, abs/2106.10199.

Hierarchy Builder: Organizing Textual Spans into a Hierarchy to Facilitate Navigation

Itay Yair[†] and Hillel Taub-Tabib[‡] and Yoav Goldberg^{†,‡}

[†]Computer Science Department, Bar Ilan University

[‡]Allen Institute for Artificial Intelligence

Abstract

Information extraction systems often produce hundreds to thousands of strings on a specific topic. We present a method that facilitates better consumption of these strings, in an exploratory setting in which a user wants to both get a broad overview of what’s available, and a chance to dive deeper on some aspects. The system works by grouping similar items together, and arranging the remaining items into a hierarchical navigable DAG structure. We apply the method to medical information extraction.

1 Introduction

We are dealing with the question of organising and displaying a large collection of related textual strings. The need arises, for example, in information extraction or text mining applications, that extract strings from text. Consider a system that scans the scientific literature and extracts possible causes for a given medical condition. Such a system may extract thousands of different strings, some of them relate to each other in various ways,¹ and some are distinct. Users consume the list in an exploratory mode (Agarwal and Sahu, 2021)(White and Roth, 2008), in which they do not have a clear picture of what they are looking for, and would like to get an overview of the different facets in the results, as well as to dig deeper into some of them.

For example, distinct strings extracted as causes for sciatica include “*herniated disc*”, “*herniated disk*”, “*lumbar disk herniation*”, “*posterior intervertebral disc herniation*” and “*endometriosis*”, among hundreds of others. The user of this system likes to go over the returned list to learn about possible causes, but going over hundreds to thousands of results is mentally taxing, and we would like to reduce this effort. In the current case, we would certainly like to treat the first two items (*herniated disc* and *herniated disk*) as equivalent and show

them as one unified entry. But we would also like to induce an additional hierarchy. For example, it could be useful to separate all the *herniated disc* related items (or even all the *disc* related items) in one branch, and the *endometriosis* case in another. This will allow the user to more efficiently get a high level overview of the high-level represented topics (*disc herniation* and *endometriosis*) and to navigate the results and focus on the cases that interest them in the context of the query (for example, they may feel they know a lot about disc-related causes, and choose to ignore this branch).

An additional complication is that the hierarchy we are considering is often not a tree: a single item may have two different parents, resulting in a direct acyclic graph (DAG). For example, arguably a condition like *leg pain* should be indexed both under *leg* (together with other leg related items) and under *pain* (together with pain related items). The hierarchy structure is contextual, and depends on the data: if there are not many other leg related items, it may not be beneficial to introduce this category into the hierarchy.

Additionally, note that some items in the hierarchy may not directly correspond to input strings: first, for the “*leg pain*” example above, if the input list does not include stand-alone *leg* or *pain* items, we may still introduce them in our hierarchy. We may also introduce additional abstraction, for example we may want to group “*heart disease*”, “*ischemia*”, “*hypotension*”, and “*bleeding*” under “*cardiovascular disease*”.

In this work we introduce a system that takes such a flat list of related strings, and arranges them in a navigable DAG structure, allowing users to get a high level overview as well as to navigate from general topics or concepts to more specific content by drilling down through the graph. Ideally, the graph would allow the user to:

(1) get a comprehensive overview of the the various facets reflected in the results;

¹Figure 1 lists the kinds of relations between strings.

Relation between spans	Example
Paraphrases: mean the same thing in different words.	'herniation of a lumbar disc', 'lumbar disc herniation'
Close meaning: similar meanings but not exactly paraphrased.	'acute myocardial infarction', 'myocardial infarction'
Elaboration: one span elaborates on the other.	'disc herniation' → 'intervertebral disc herniation' → 'posterior intervertebral disc herniation'
Different: describe entirely different concepts.	'disc herniation' vs 'spinal stenosis'
Partially different: describe more than one more general concept	'stenosis of the lumbar spinal canal' → 'lumbar spinal stenosis', 'spinal canal stenosis'
Co-mention: both mention the same concept, but do not elaborate on it.	'tumor in the leg ', ' leg pain' 'brain tumor ', ' tumor in the leg'
Taxonomical: the spans mention concepts that have a shared parent in a taxonomy.	'aspirin', 'penicillin' → 'DRUG' 'biliary tract disease', 'hepatitis' → 'HEPATOPATHY'

Figure 1: Kinds of possible relations between input strings

(2) quickly get an overview of main aspects in the results;

(3) efficiently navigate the results, finding items in the sub-graph in which they expect to find them.

At a high level, the system works by finding lexically equivalent terms, arranging them in a DAG structure reflecting the specificity relation between terms, further merging equivalent nodes based on a neural similarity model, add additional potential intermediary hierarchy nodes based on taxonomic information and other heuristics, and then pruning it back into a smaller sub-DAG that contains all the initial nodes (input strings) but only a subset of the additional hierarchy nodes. Finally, we select the top-k “entry points” to this graph: high level nodes that span as many of the input nodes as possible. This process is described in section §3. While the DAG extended with potential hierarchies is very permissive and contains a lot of potentially redundant information, the DAG pruning stage aims to ensure the final graph is as compact and informative as possible.

We focus on causes-for-medical-conditions queries, and provide a demo in which a user can select a medical condition, and browse its causes in a compact DAG structure.

To evaluate the resulting DAGs, we perform automatic and manual evaluation. The automatic evaluation is based on measuring various graph metrics. The human evaluation is performed by human domain experts. Our results show that the DAG structure is significantly more informative and effective

than a frequency-ranked flat list of results.

2 Requirements

As discussed in the introduction, our input is a list of strings that reflect answers to a particular question, as extracted for a large text collection (we focus in this paper on the biomedical domain, and more specifically in causes for medical conditions). This list can be the output of an Open-IE system (Fader et al., 2011; Stanovsky et al., 2015; Kolluru et al., 2020), the results of running extractive QA (Rajpurkar et al., 2016) with the same question over many paragraphs, or extracted using an extractive query in a system like SPIKE (Shlain et al., 2020; Taub Tabib et al., 2020; Ravfogel et al., 2021). The lists we consider typically contain from hundreds to thousands of unique items. We identified a set of relations that can hold between strings in our inputs, which are summarized in Table 1. We would like to arrange these items in a hierarchical structure to facilitate exploration of the result list by a user, and allow them to effectively consume the results. Concretely, the user needs to:

- a. *not see redundant information.*
- b. *be able to get a high-level overview of the various answers that reflected from the results.*
- c. *be able to get a quick access to the main answers.*
- d. *be able to dig-in into a specific phenomenon or concept that is of interest to them.*

e. be able to locate concepts they suspect that exist.

This suggests a hierarchy that respects the following conditions:

Paraphrased spans should be combined into a single group, and *close-meaning* spans should be combined into the same group; *Elaboration* relations should be expressed hierarchically; *Co-mention* spans should be both descendants of the shared concept; *Taxonomic relations* should (in some cases) be descendants of the taxonomical parent.

Additionally, we would like each node in the hierarchy to have relatively few children (to reduce the need to scan irrelevant items), yet keep the hierarchy relatively shallow (to save expansion clicks if possible). The hierarchical structure should also be informative: we should be able to guess from a given node which kinds of items to expect to find under it, and which kinds of items *not* to expect to find under it. This means a single item should be lockable in different ways, in case it can be categorized under different keys (we would sometimes like “*brain tumor*” to be listed under *brain* and sometimes under *tumors*).²

3 Method

Expanding the initial list. We assume that the strings in the initial list are *maximal*, meaning that the string captures the extracted noun-phrase including all of its possible modifiers. We further expand the list by considering also potential substrings of each maximal string, reflecting different granularities. For example, from the string “severe pain in the lower right leg” we would extract “pain”, “severe pain”, “severe pain in the leg”, “severe pain in the lower right leg”, among others.³ We then consider the union of the initial set of input strings and the set of additional sub-strings. Different users would be interested in different granularities depending on their information need. We rely on the DAG-pruning stage to properly organize these strings and prune away non-informative ones in the context of the entire set.

Initial grouping into equivalence sets. The input of this stage is a set of strings (the union of the input set and the extended set), and the output is a

²Arranging information as graphs to facilitate navigation and exploration is, of course, not a novel concept. A notable example is entailment graphs (Kotlerman et al., 2015; Adler et al., 2012).

³This is done using a rules-based algorithm that operated on the parse tree, which extracted all the distinct modification spans derived from the head token.

list of sets, such that the sets are distinct, and their union covers the initial set. For example, after this stage, the items “*herniated disk*”, “*herniated disc*”, “*disc herniation*”, “*herniation of the disc*” will be in the same equivalence set.

The grouping in this stage is inspired by (Gash-teovski et al., 2017) and is based on considering each string as a bag of lemmas, discarding stop words, modal words, and quantity words, and considering items as equivalent if their bags are equivalent. The lemma matching is relaxed, and allows, beyond exact string match, also matches with small edit distance and matches based on UMLS (Bodenreider, 2004) and WordNet (Miller, 1992) spelling variants and synonyms.

Initial DAG construction. We now take the list of sets from the previous stage, and arrange them into a DAG, where each set is a DAG node. We add a directed edge between two nodes A and B if B is more specific than A, and no other node C is more specific than A and less specific than B.

The *specificity relation* at this stage is determined based on the bags of lemmas that were used to create the equivalence sets: a set B is more specific than a set A if A and B are not equivalent and the bag of B contains the bag of A.

Adding heads as nodes For all spans, we take their head-word (either a single adjective or a single noun) and add them as roots of the DAG. We then add an additional root node above them, so that the DAG has a single root. This handles the co-mention relation.

Merging semantically equivalent graph nodes. We now take the DAG and merge equivalent nodes, as determined by a trained statistical model (we use SAP-BERT (Liu et al., 2020))⁴. For example, this stage will merge “*administration of streptozotocin*” and “*streptozotocin injection*”. When merging two graph nodes, we handle the corresponding edges in the expected way (the children of the two individual nodes become children of the merged node, and the parents of the individual nodes become the parents of the merged node).⁵

⁴We chose SAP-BERT for its entity-linking specialization, and since it outperformed other models we tried, such as SciBERT (Beltagy et al., 2019), in detecting semantic similarity for our specific case.

⁵We perform this stage after the DAG construction and not prior to it, as it makes the specificity relation between nodes significantly harder to define. In the current order, we first define specificity based on lexical containment, and then add further merge the groups.

For a pair of graph nodes A and B, we encode each string in A and in B into a vector using SAP-BERT, and represent each node as the average vector of the strings within it. We go over the nodes in the DAG in DFS order starting from the root nodes, and for each node consider all of its children for potential merging among them. We merge two nodes if the cosine similarity score between their vectors passes the threshold $t_1 = 0.9$ and their merging does not create a cycle. We then do another pass and merge nodes to direct child nodes if their similarity score is above $t_2 = 0.95$, again avoiding creating circles.

After this stage, we attempt to further merge nodes based on the UMLS ontology (Bodenreider, 2004). Two nodes A and B are considered UMLS-equivalent, if there is at least one string in node A that is listed in UMLS as a synonym of at least one string in node B. Such cases are merged.⁶

Adding taxonomic nodes. So far the relationships between nodes in the DAG were solely based on lexical relations. In order to enrich the graph, we introduce additional nodes based on taxonomical relations, which are not reliant on lexical information. For instance, “heart disease”, “ischemia”, “hypotension”, and “bleeding” are under the broader term “cardiovascular disease”. We add many nodes here, relying on many of them to be pruned in the next stage.

We map each node to the UMLS hierarchy, and look for UMLS concepts that govern at least two DAG nodes (“descendent DAG nodes”). These are potential abstractions over graph nodes. For each such UMLS concepts that is already part of the DAG, it is connected by an edge to all its descendant DAG nodes that do not already have a path to them, if adding such an edge does not create a cycle. For UMLS concepts that are not already in the DAG, they are added as new nodes governing the descendant graph nodes. UMLS concepts have multiple synonyms. When adding them as nodes, we choose the synonym with the highest SAP-BERT cosine similarity to the descendent DAG nodes this concept governs.

DAG Pruning. The DAG at this stage is quite large and messy, containing both nodes containing input strings, as well as additional hierarchy nodes based on linguistically motivated substrings of the input strings, and on taxonomic relations.

⁶If this merging creates a cycle, this cycle is removed.

We prune it to create a smaller graph which is more amenable to navigation. The smaller DAG should contain all the nodes corresponding to input strings, and an effective set of additional hierarchy nodes. Some of the hierarchy nodes are more important than others, as they provide a better differential diagnosis among the answers. Our goal is to highlight these and filter out the less important ones. Operatively, we would like for each node in the graph to have the minimal number of children, such that all the input strings that were reachable from it, remain reachable from it. This focuses on hierarchy nodes that are shared among many input concepts. We first prune graph edges according to this criteria. This process result in nodes that have a single child. Such nodes are removed, and their children are attached to their parent.⁷ Selecting the minimal number of children according to this criteria is NP-hard. As an alternative, we use an approximation algorithm called the greedy set cover algorithm (Johnson, 1973), which works by selecting in each step the node with the highest number of non-covered answers, covering them, and proceeding. This helps in choosing the most important concepts and with the highest differential diagnosis.

Entry-point selection. Finally, we seek k nodes that will serve as the “entry nodes” to the graph. These should be k nodes that fulfill the following criteria:

- allow reaching as many input strings as possible.
- the semantic affinity between a node and the input string reachable by it, is high.

The users will initially see these nodes as well as an additional “other” node, from which all the other input strings can be reached. The entry node labels provide an overview of the k main concepts in the list, and allow the user to both get an overview of the result as well as to drill down into parts that interest them. Criteria (b) is important to ensure that the user not only can reach the input string by navigating from an entry point, but also that it will *expect* to find this input string there.

This selection is done by a heuristic algorithm which we adapted from the Greedy+ DAG-node-selection algorithm in (Zhu et al., 2020). It first assigns each node C with a score that combines the

⁷Selecting the smallest group of concepts at each hierarchy level is important for user navigation, who quickly become overwhelmed by too many nodes, making it difficult to orient themselves within the DAG.

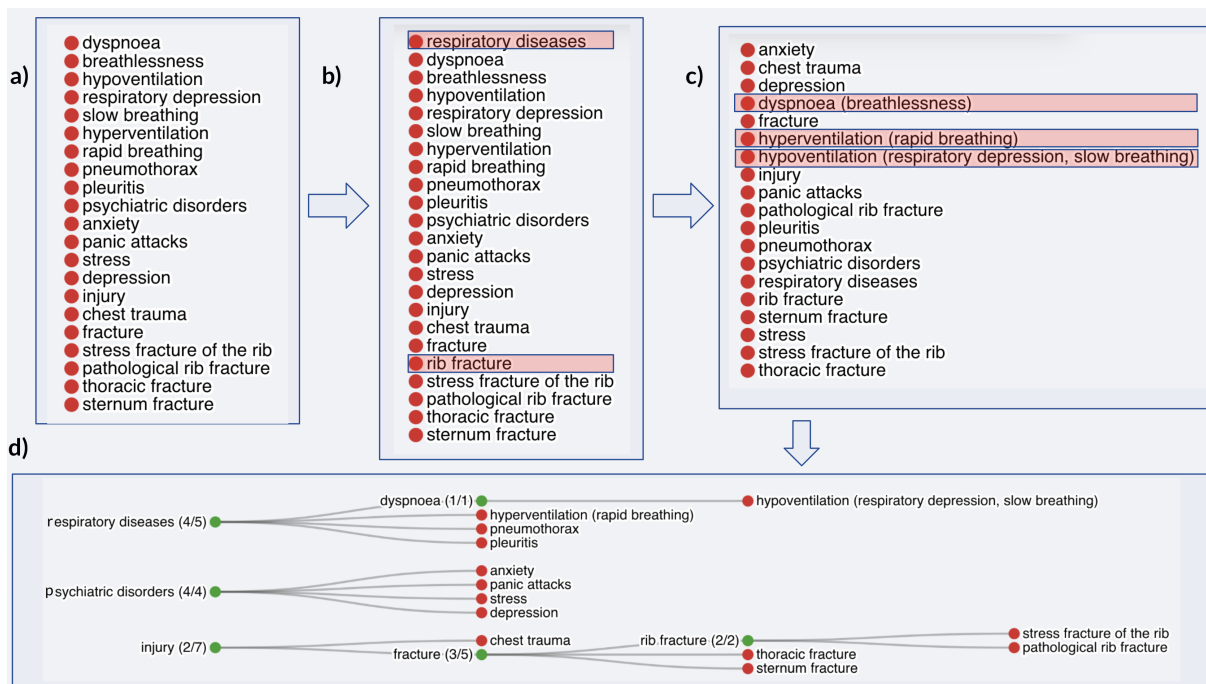


Figure 2: Input-Output Example. See section §4.

number of the input nodes reachable from it, and the semantic affinity (based on SAP-BERT cosine similarity) of C to each of these reachable nodes. It then iteratively adds the highest scoring candidate C to the set of entry points, and adjusts the scores of each remaining node N by subtracting from the score of N the affinity scores between C and the input nodes reachable from N. We do this until we reach k entry points.

Visualization. We are now finally ready to show the DAG to the user. For nodes that correspond to multiple (semantic equivalent but lexically different) input strings, we choose one of them as the representative for display purposes.

4 Input-output Example

We demonstrate with a minified example. Given the set of spans in Figure (2a), representing causes of chest pain, Hierarchy Builder expands the set by adding the spans "rib fracture" (this is a substring of two existing spans) and "respiratory diseases" (a new taxonomic node). Based on the expanded set of spans in Figure (2b), Hierarchy builder identifies synonymous spans and merges them into the concepts. In Figure (2c) we see these concepts, where each concept includes aliases in parenthesis where applicable. Hierarchy Builder then places the entries in a DAG based on a hierarchy of specificity, as depicted in Figure (2d).

5 Experiments and Evaluation

Scope We focus on the medical domain and evaluate our system on etiologies (causes) of two medical symptoms (“*jaundice*” and “*chest pain*”). These symptoms were chosen because they are common and each contain many different etiologies mentioned in the literature.

The input lists for the system were the result of running a set of 33 syntactic patterns over PubMed abstracts, looking for patterns such as “COND due to ___” or “patients with COND after ___” where COND is either *jaundice* or *chest pain*. The results were extracted using the SPIKE system (Shlain et al., 2020; Taub Tabib et al., 2020) and each matched head-word was expanded to the entire syntactic subgraph below it. This resulted in 3389 overall extracted strings and 2623 unique strings for *jaundice* and 2464 overall and 2037 unique for *chest pain*. After merging strings into synonym sets as described in §3, we remain with 2227 concepts for *jaundice* and 1783 for *chest pain*.

For each of the symptoms there are established and widely accepted lists of common etiologies, which we rely on in our evaluation.⁸ We take 38 established etiologies for *jaundice* and 33 for *chest*

⁸We take the established etiologies for *jaundice* from <https://www.ncbi.nlm.nih.gov/books/NBK544252/> and for *chest pain* from <https://www.webmd.com/pain-management/guide/whats-causing-my-chest-pain>.

pain, and check their accessibility in the flat list of extracted symptoms, as well as in the hierarchical DAG we create.

Coverage and Entry-point Selection For *jaundice*, our input list contains 28 out of the 38 known etiologies, and for *chest pain* 26/33. With $k = 50$, 25 of 28 concepts are reachable from an entry point for *jaundice* and 21/26 for *chest pain*. With $k = 100$ the numbers are 28/28 (*jaundice*) and 24/26 (*chest pain*).

Assessing the contribution of the different components The different components in our algorithm contribute by adding nodes, combining nodes, adding edges, and removing edges. Table 1 describes the kind of contribution of each component and quantifies its impact, for each of the two tested conditions.

We now look at the case where we select 50 entry-point nodes, and focus on the effect on the top-level nodes. We see that for Chest-pain, a total of 20 of the 50 selected entry-points were not in the original input, but were added by the various components (12 from expanding the initial list, 5 from adding head words, and 3 from taxonomic words). Similarly, for Jaundice, these components added a total of 29 root nodes (out of the selected 50) that were not in the original input (17 from expanding initial list, 5 from head words and 6 from taxonomic nodes).

The “Expanding the initial list” component plays a significant role in shaping the DAG structure. In Chest Pain, 161 out of 224 internal nodes originate from the expanded list (146 from Expanding the initial list and 15 from co-mention). In Jaundice, 347 out of 423 internal nodes stem from the expanded list (333 from Expanding the initial list and 14 from co-mention). This highlights the substantial impact of this component on the DAG’s structure.

The number of merges performed indicates the usefulness of the employed merging methods.

Furthermore, the set cover pruning algorithm effectively reduces the number of edges in the DAG.

Qualitative Measures For *jaundice*, our final DAG contains 2620 nodes overall and has a maximum depth of 11. With $k = 50$ The average number of leaves per entry point is 22.68 (min 0, max 600), and the average depth is 2.86 (min 0, max 9). Most importantly, each internal node has an average of 9.12 children (min 1, max 56, variance 34.91), making them highly browsable.

For *chest pain*, the trends are overall similar: our final DAG contains 2124 nodes overall and has a maximum depth of 9. With $k = 50$ The average number of leaves per entry point is 14.14 (min 1, max 175), and the average depth is 2.8 (min 0, max 7). Each internal node has an average of 4.94 children (min 1, max 53, variance 27.53).

Human evaluation. Our main evaluation centers around the effort for an expert⁹ to locate the known etiologies in the resulting DAG, compared to a flat list sorted by frequency. For each of the etiologies, we ask how many entries need to be considered before finding the etiologies. For the flat list, this means how many items are read when scanning the list in order before reaching the etiology. For the DAG, we count the number of clicks (expansions of a node) starting from $k = 50$ entry points (a quantity that aligns with a reasonable threshold of entry nodes perceivable by a user), while summing also the number of items before the expanded node in each level. Note that since we look for common etiologies rather than rare ones, we would assume a frequency-ranked list based on literature mentions would compare favorably in these measures. Nonetheless, we see a clear benefit of the DAG. We compare to conditions: an ideal condition where the user knows exactly which nodes to expand (blue in the graph), and a realistic scenario, in which the user searches for the etiologies by expanding nodes (gray in the graph).

We also perform another evaluation in which we ask the experts to rank each path to an etiology based on its quality, given the question “to what extent is this a logical path to follow in order to find the etiology”, on a scale of 1 (very bad) to 5 (very good).

Results Figure 3 shows the main results for the two conditions. Despite the frequency-based ranking, many of the etiologies appear relatively low in the flat list, making them very hard to come by in this condition (orange). On the other hand, when considering the DAG, the vast majority of items are significantly easier to locate, requiring scanning significantly fewer items. Only 3 items for jaundice and 2 for chest pain were significantly harder to locate in the DAG than in the flat list. In terms of the quality of the DAG paths associated with each

⁹We use two experts, each evaluating a different condition. The expert evaluating *jaundice* is an expert MD specializing in children’s medicine. The expert evaluating *chest pain* is a PhD in biology with 38 years of biomedical research.

Component	Contribution	Chest-pain	Jaundice
Expanding the initial list (for full DAG)	Add nodes	504	893
Expanding the initial list (for DAG with 50 entry nodes)	Add nodes	158 (12 top level)	350 (17 top level)
Adding heads as nodes (Full DAG)	Add nodes	457	379
Adding heads as nodes (50 entry nodes)	Add nodes	20 (5 top level)	19 (6 top level)
Merging semantically equivalent nodes	Merge nodes	93 (out of 2556)	266 (out of 3330)
UMLS merging of synonym nodes	Merge nodes	62 (out of 2504)	99 (out of 3167)
UMLS taxonomic nodes (full DAG)	Add nodes	113	169
UMLS taxonomic nodes (50 entry nodes)	Add nodes	3	6
UMLS taxonomic edges	Add edges	140 (5 top level)	153 (3 top level)
DAG Pruning	Remove edges	2363	3209

Table 1: Quantifying the contribution of the different components.

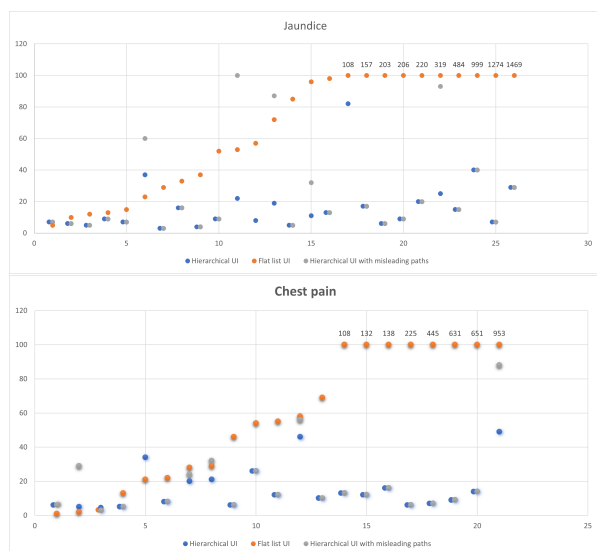


Figure 3: Effort to reach a set of common etiology items using our created DAG vs. a frequency ranked list. X axes coordinates correspond to different etiologies sorted by their frequency in the input list, and Y axes corresponds to effort. Orange: frequency-ranked flat list. Blue: DAG + oracle locating of items. Gray: DAG + human locating of items.

etiology, the jaundice annotator ranked 23 out of 25 as 5, 1 as a 2, and 1 as a 1. For chest pain, the numbers are 19 out of 21 ranked as 5, 1 as 2, and 1 as 1. Overall, our hierarchy building algorithm works well for the vast majority of the cases, and offers significant benefits over the flat list.

6 Conclusions

We presented an automatic method to organize large lists of extracted terms (here, of medical etiologies) into a navigable, DAG-based hierarchy, where the initial layer provides a good overview

of the different facets in the data, and each internal node is has relatively few items. The code together with a video and an online demonstration are available at <https://github.com/itayair/hierarchybuilder>.

7 Limitations

While our method is aimed at organizing any flat-list of extractions, we evaluated it here only on the medical domain, only on a single kind of information need (etiologies), and only for common conditions (jaundice and chest pain). More extensive evaluation over additional conditions is needed in order to establish general-purpose utility. However, we do find the system useful for navigating in automatically-extracted etiology lists, and encourage the readers to experiment with the system also on other conditions, to assess its utility.

There are also some candidates for improving the method also in the biomedical domain, which are not currently handled: (a) abstraction over substrings. e.g., for the spans “*administration of penicillin*”, “*administration of aspirin*”, “*administration of augmentin*”, it could be useful to introduce an shared parent level of “*administration of antibiotic/drug*”. Our system can currently identify *penicillin*, *augmentin*, *aspirin* as an *antibiotic/drug*, but cannot handle abstraction over sub-strings. (b) Linking to UMLS currently relies on exact lexical matches, and can be improved.

8 Ethical Considerations

We present a system for organizing large result lists into a browsable hierarchy. In general, consuming a hierarchy is more effective than consuming a very

long list. However, hierarchies can hide items, especially if the items are misplaced in an unexpected branch—which our system sometimes does (albeit rarely). In situations where consuming the entire information is crucial and the cost of missing an item is prohibitive or dangerous, a flat list would be the safer choice.

Acknowledgements This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme, grant agreement No. 802774 (iEXTRACT).

References

- Meni Adler, Jonathan Berant, and Ido Dagan. 2012. [Entailment-based text exploration with application to the health-care domain](#). In *Proceedings of the ACL 2012 System Demonstrations*, pages 79–84, Jeju Island, Korea. Association for Computational Linguistics.
- Manoj K Agarwal and Tezan Sahu. 2021. Lookup or exploratory: What is your search intent? *arXiv preprint arXiv:2110.04640*.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, pages D267–D270.
- 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*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. [MinIE: Minimizing facts in open information extraction](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, Copenhagen, Denmark. Association for Computational Linguistics.
- David S Johnson. 1973. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49.
- Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. 2020. [OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3748–3761, Online. Association for Computational Linguistics.
- Lili Kotlerman, Ido Dagan, Bernardo Magnini, and Luisa Bentivogli. 2015. Textual entailment graphs. *Natural Language Engineering*, 21(5):699–724.
- Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2020. Self-alignment pretraining for biomedical entity representations. *arXiv preprint arXiv:2010.11784*.
- George A. Miller. 1992. [WordNet: A lexical database for English](#). In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.
- 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.
- Shauli Ravfogel, Hillel Taub-Tabib, and Yoav Goldberg. 2021. [Neural extractive search](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 210–217, Online. Association for Computational Linguistics.
- Micah Shlain, Hillel Taub-Tabib, Shoval Sadde, and Yoav Goldberg. 2020. [Syntactic search by example](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 17–23, Online. Association for Computational Linguistics.
- Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. [Open IE as an intermediate structure for semantic tasks](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 303–308, Beijing, China. Association for Computational Linguistics.
- Hillel Taub Tabib, Micah Shlain, Shoval Sadde, Dan Lahav, Matan Eyal, Yaara Cohen, and Yoav Goldberg. 2020. [Interactive extractive search over biomedical corpora](#). In *Proceedings of the 19th SIG-BioMed Workshop on Biomedical Language Processing*, pages 28–37, Online. Association for Computational Linguistics.
- Ryen W White and Resa A Roth. 2008. Evaluation of exploratory search systems. In *Exploratory Search: Beyond the Query—Response Paradigm*, pages 61–69. Springer.

Xuliang Zhu, Xin Huang, Byron Choi, and Jianliang Xu.
2020. Top-k graph summarization on hierarchical
dags. In *Proceedings of the 29th ACM International
Conference on Information & Knowledge Manage-
ment*, pages 1903–1912.

CARE: Collaborative AI-Assisted Reading Environment

Dennis Zyska*, Nils Dycke*, Jan Buchmann, Iliia Kuznetsov, Iryna Gurevych

Ubiquitous Knowledge Processing Lab (UKP Lab)

Department of Computer Science and Hessian Center for AI (hessian.AI)

Technical University of Darmstadt

ukp.informatik.tu-darmstadt.de

Abstract

Recent years have seen impressive progress in AI-assisted writing, yet the developments in AI-assisted *reading* are lacking. We propose *inline commentary* as a natural vehicle for AI-based reading assistance, and present CARE: the first open integrated platform for the study of inline commentary and reading. CARE facilitates data collection for inline commentaries in a commonplace collaborative reading environment, and provides a framework for enhancing reading with NLP-based assistance, such as text classification, generation or question answering. The extensible behavioral logging allows unique insights into the reading and commenting behavior, and flexible configuration makes the platform easy to deploy in new scenarios. To evaluate CARE in action, we apply the platform in a user study dedicated to scholarly peer review. CARE facilitates the data collection and study of inline commentary in NLP, extrinsic evaluation of NLP assistance, and application prototyping. We invite the community to explore and build upon the open source implementation of CARE¹.

1 Introduction

Individual and collaborative text work is at the core of many human activities, including education, business, and research. Yet, reading text is difficult and takes considerable effort, especially for long and domain specific texts that require expert knowledge. While past years have seen great progress in analyzing and generating text with the help of AI – culminating in strong generative models like GPT-3 (Brown et al., 2020) and ChatGPT (Ouyang et al., 2022)² – the progress in applications of AI to reading and collaborative text work is yet to match these achievements. The ability of modern generative models to create natural-sounding but factually

*These authors contributed equally to this work

¹<https://github.com/UKPLab/CARE>

²<https://openai.com/blog/chatgpt>

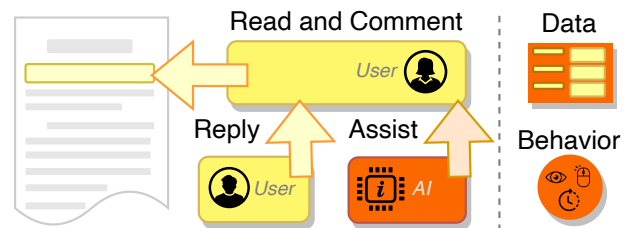


Figure 1: CARE allows users to collaboratively read and discuss texts, provides a generic interface for AI-based reading assistance, and collects research-ready textual and behavioral data.

flawed outputs (Ji et al., 2022) stresses the need for supporting humans in critical text assessment.

Humans use annotations to read and collaborate over text, from hand-written print-out notes to highlights in collaborative writing platforms. This makes in-text annotations – *inline commentaries* – a promising vehicle for AI-based reading assistance. For example, an AI assistant could automatically classify the user’s commentaries, or verify and provide additional information on the highlighted passages. Yet, the lack of data and key insights limits the NLP progress in this area: from the foundational perspective, we lack knowledge about the language of inline commentaries, as most of this data is not openly available for research. From the applied perspective, little is known about the hands-on interactions between humans and texts, how they translate into NLP tasks, and how the impact of NLP-based assistance on text comprehension can be measured. While ethical, controlled data collection has been receiving increasing attention in the past years (Stangier et al., 2022), data collection tools for inline commentary are missing, and so are the tools for applying and evaluating NLP models within a natural reading environment.

To address these limitations, we introduce CARE: a Collaborative AI-Assisted Reading Environment, where users can jointly produce in-

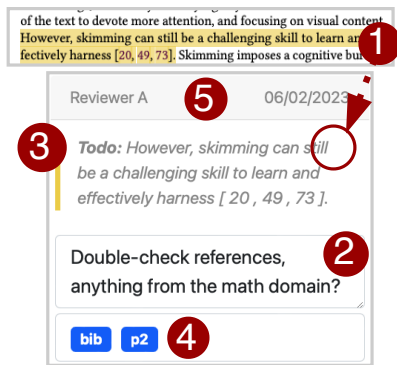


Figure 2: An inline commentary in CARE consists of a highlight (1) optionally associated with a commentary text (2), a label (3), a number of free-form tags (4) and metadata (5), e.g. user name and creation time.

line commentaries on PDF documents in an intuitive manner, connected to a model-agnostic and flexible AI assistance interface. Unlike existing labeling tools, CARE provides a (1) familiar, task-neutral environment for collaborative reading similar to the tools used in everyday text work; unlike off-the-shelf reading and writing applications, CARE offers (2) structured machine-readable data export functionality, including both inline commentary and behavioral data; unlike task-specific AI-assisted reading tools, CARE features a (3) generic interface for integrating NLP modules to support reading and real-time text collaboration.

Our contribution has multiple audiences. For NLP researchers, CARE makes it possible to efficiently collect inline commentary data in a standardized manner, and provides a generic interface for the extrinsic evaluation of NLP models. For application designers, CARE offers an extensible platform and behavioral metrics to study how humans interact with texts. For users, CARE enables the development of new, innovative applications built around AI-assisted reading such as interactive e-learning, community-based fact-checking, and research paper assessment. To foster the progress in AI-assisted reading research, we make the implementation openly available and easy to deploy and to extend.

2 Background

2.1 Terminology and Requirements

The term "annotation" allows for broad interpretation and encompasses both the results of controlled annotation studies that enrich text with a specific new information layer (e.g. named entities), and

the less-regulated, natural annotations that humans produce when they work with text. Yet, the two annotation mechanisms are fundamentally different. Labeled NLP data is usually obtained via annotation studies – supervised campaigns that involve formalized tasks and labeling schemata, detailed guidelines, and are supported by specially designed annotation software that requires training of the annotators. However, collecting natural annotation data requires the opposite: the process should minimally interfere with the user’s workflow, and the tool should provide a natural environment for working with text given the task at hand. Our work addresses annotation in the latter sense. To avoid ambiguity, we propose the term *inline commentary* to denote in-document highlights left by the users while reading and collaborating on text, potentially associated with commentary text, tags and metadata (Figure 2). We reserve the term *labeling* for the traditional NLP markup.

With this distinction in mind, for a tool to support the study of inline commentary we define the following **requirements** distributed among the key USER GROUPS:

A. Natural environment: The tool should provide the READER with a natural reading environment, specified as allowing the READER to (A1) leave inline commentaries on the documents in (A2) common reading formats like PDF, while requiring (A3) minimal to no training.

B. Collaboration: The tool should run (B1) online and support (B2) real-time collaboration where the READERS can leave, see and reply to each others’ commentaries in an on-line fashion.

C. Data management: The tool should enable RESEARCHERS, APPLICATION DEVELOPERS and ADMINISTRATORS to easily (C1) import new documents, (C2) collect inline commentary and USER behavior data, and (C3) export this data in a machine-readable format for further scrutiny.

D. Openness and extensibility: Both documents and inline commentaries might contain confidential data. It is thus crucial that a tool can be (D1) self-hosted on-premise and allows controlling user access to the data. AI-assisted reading has many potential use cases, stressing the need for (D2) high configurability and easy deployment of the tool. To promote transparency and facilitate extension, the platform should be available as (D3) open-source.

E. AI assistance: Finally, the tool should provide an easy way to (E) integrate AI assistance modules

for RESEARCHERS and DEVELOPERS to support USERS in reading and comprehending text.

2.2 Related tools

We identify four broad groups of software tools falling within the scope of our requirements, which we briefly exemplify below. Our overview demonstrates the wide use of inline commentary "in the wild" and underlines the limitations of the existing solutions for the systematic study of inline commentary and AI-assisted reading in NLP.

Readers Highlighting and inline commentary are core features of most standalone reading tools, from PDF viewers like Adobe Acrobat Reader³ to literature management software like Mendeley⁴. The most commonly used tools are proprietary and thereby hard to extend, and do not offer management, collection and export of fine-grained data, making them unsuitable for the study of inline commentary. While a few dedicated reading applications like ScholarPhi (Head et al., 2021), Scim (Fok et al., 2022), SciSpace⁵, and Scrible⁶ do offer machine-aided reading assistance, they focus on their particular use cases, lack data collection functionality and extensibility, and can not be easily hosted on-premise to protect potentially sensitive user and document data.

Social annotation Focusing on the collaborative aspect of reading, social annotation platforms allow users to exchange their inline commentaries via a centralized platform. A prime example is Hypothes.is⁷, which offers a natural environment, is available open-source and provides a standardized mechanism for exporting inline commentary. Yet, the platform is not easy to extend and customize, and does not offer a standardized mechanism for integrating AI-assistance or behavioral data collection. While not being based on Hypothes.is, CARE adopts many of its design ideas, including the appearance and functionality of the annotation sidebar, utilities to locate inline commentaries in the document text, as well as the underlying data structure of the annotations.

Authoring tools Inline commentary is featured in many text authoring tools, from standalone of-

fice applications like Microsoft Office⁸ to collaborative web-based platforms like Google Docs⁹ and Overleaf¹⁰. While widely used and familiar, these applications are hard to tailor to the needs of a particular scientific study, offer limited data export capabilities, lack flexible AI integration for assistance, and are either implemented as standalone desktop applications (impeding real-time collaboration), or do not allow self-hosting, making ethical data collection and storage challenging.

Labeling tools The rapid progress in NLP of the past decades has been accompanied by the evolution of general-purpose tools used to acquire labeled data (Neves and Ševa, 2019), from early desktop applications like *WordFreak* (Morton and LaCivita, 2003) to modern extensible, web-based, open-source environments like *brat* (Stenetorp et al., 2012), *labelstudio*¹¹, *docanno*¹² and *INCEpTION* (Klie et al., 2018). CARE inherits many concepts from NLP annotation platforms – including coupling of external recommenders (Klie et al., 2018), tag sets and study management functionality, and flexible data export. Although not specifically designed for controlled labeling scenarios, CARE can be used as a lightweight labeling tool with collaboration and assistance capabilities.

3 Platform Description

CARE addresses the gap in existing solutions that prevents the study of inline commentary and AI-assisted reading. Here we review the main components of CARE from the user perspective, while the next Section outlines the key technical aspects of our open implementation. We discuss the components in order of importance and refer to the Appendix A for the illustration of a typical user journey.

At the core of CARE is the **reading component** which allows users to attach inline commentaries to documents. To ensure that the visual representation of the document remains true to its source and stable across platforms, CARE focuses on PDF as the main source format¹³. An inline commentary can amount to a simple highlight attached to a continuous text span, can be associated with a free-text

³<https://www.adobe.com/acrobat/pdf-reader.html>

⁴<https://www.mendeley.com>

⁵<https://typeset.io>

⁶<https://www.scribble.com>

⁷<https://web.hypothes.is>

⁸<https://www.office.com/>

⁹<https://www.google.com/docs/about>

¹⁰<https://www.overleaf.com>

¹¹<https://labelstud.io>

¹²<https://github.com/doccano/doccano>

¹³While support for other document formats is planned, we note that any textual document can be converted into a PDF.

note, and can carry a label from a pre-configured label set, as well as any number of free-form tags (Figure 2). It is possible to add document-level commentaries that are not attached to a span. In-line commentaries are displayed in the dedicated **CARE sidebar** and can be navigated and edited. The process is **collaborative**: multiple users can leave inline commentaries on the same document and reply to them in real time. The commentaries are saved and can be revisited at a later point; the resulting data can be **exported** in an easy-to-use data format, individually or in aggregate, and displayed within the user interface of CARE. In addition to the textual data, CARE collects and exports basic behavioral metrics; for instance, the time of highlight creation and the users' scrolling behavior within the document.

The second key component of CARE is **AI assistance**: the inline commentary data can be routed to an arbitrary external NLP module, which returns the prediction that can be displayed in the annotation component *in close-to-real-time* as labels, inline commentary replies, or via a custom UI. The interaction between users and AI assistance is mediated by a flexible **broker** system that distributes the processing tasks among a set of NLP models. Multiple AI assistance model instances can be acting simultaneously, and the pool of models can be extended easily through registering a new model node at the broker backend. At the moment of writing, CARE provides examples to supports integration of any pre-trained model compatible with the *huggingface transformers* API (Wolf et al., 2019) by simply changing the configuration parameters. The model then has access to the inline commentary text, highlighted span from the main document, labels, tags and metadata. It is possible to adapt CARE to use models based on other frameworks.

Finally, CARE features a flexible and configurable **dashboard** that provides quick access to user and system settings, document and label set management, and study management. In particular, the **user management** component is responsible for registration, authentication and authorization; to encourage responsible data collection and ensure that the collected inline commentary data can be used in research, CARE features sample **informed consent** forms that users are presented upon registration, along with the necessary **licensing disclaimers**, which can be refined by the study administrator.

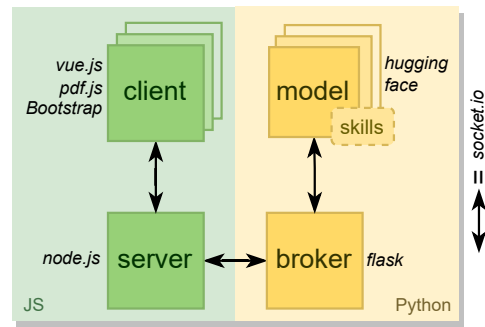


Figure 3: Overview of CARE system architecture.

4 System Design

CARE is designed to be generic, modular and extensible (Figure 3). The ability to build and deploy CARE via a Docker container makes it easy to set it up in new environments. While CARE features detailed documentation, here we provide a high-level overview of the system design. CARE follows a client-server architecture, preferring client-side operation whenever possible to speed up execution and reduce network traffic and server load. This results in a clear separation of responsibilities between the client, the server and the NLP assistance components of CARE and affords high modularity. While the main client-server pair is purely JavaScript-based, the AI models and the broker are implemented in Python to facilitate the NLP assistance development by the natural language processing community.

The web-based **CARE client** is responsible for frontend rendering and annotation functionality. The client is fully implemented in *vue.js*¹⁴, allowing dynamic rendering, modular frontend structure and reuse of original and third-party components. *Bootstrap*¹⁵ is used throughout the frontend to ensure consistent styling and responsive design; document rendering is handled via *pdf.js*¹⁶. In addition, we adopt the localization code from *hypothesis.is*¹⁷ to locate inline commentaries in the document. The **CARE server**, in turn, is responsible for synchronizing the data among clients, authentication and authorization, and for connecting to external services, including the AI-assistance broker. In line with the JavaScript-based frontend, the backend is implemented in *node.js*¹⁸ as a cross-platform run-

¹⁴<https://vuejs.org>

¹⁵<https://getbootstrap.com>

¹⁶<https://mozilla.github.io/pdf.js>

¹⁷<https://github.com/hypothesis/client>

¹⁸<https://nodejs.org/en>

time environment. As keeping message transition time low is crucial for collaboration and AI assistance, we base all communication on the *WebSocket protocol*¹⁹. Persistent bidirectional connection between the client and server components enables real-time exchange of messages and reduces communication time to the possible minimum by reducing the number of connection setups (i.e., three-way handshakes).

AI assistance in CARE is implemented by routing user requests to separately hosted NLP models abstracted into **Skills**: high-level machine-readable specifications of assistance functionalities including inputs, outputs and model configurations (Baumgärtner et al., 2022). The current implementation of NLP assistance in CARE is built on top of the *huggingface* pipeline, making it easy to integrate a wide range of pre-trained models; we provide sample code to facilitate building self-registering docker containers for NLP model deployment. The interactions between the server and the NLP models is mediated by a **Broker** system which distributes user requests among NLP models depending on the necessary skill.

5 User Study

To evaluate and refine the reading environment of CARE in the context of a collaborative applied task (requirements A and B), and to ensure the data export functionality (C) and the extensibility (D) of the system, we have extended the base configuration of CARE to accommodate a custom reading scenario and conducted a user study. We describe the core components of the study here and refer to the Appendix B.1 for details.

Task Scholarly peer review is a prototypical example of close reading accompanied by note-taking, where an expert assesses a manuscript in terms of its originality, readability, validity and impact (Jefferson et al., 2002). We adopted critical reading that takes place during peer review as a basis for our task. The participants of the study were provided with a manuscript-to-review and instructed to leave self-contained annotations on the manuscript while reading. To incentivize reviewers to perform the task rigorously, we simulated a subsequent acceptance-decision-making phase based on the provided annotations. To support the sce-

¹⁹<https://datatracker.ietf.org/doc/html/rfc6455>

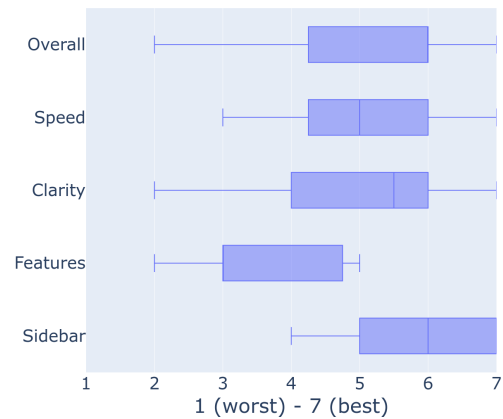


Figure 4: Usability questionnaire results.

nario, we extended CARE to allow reviewer-paper assignment and decision-making functionality.

Study design We selected two nine-page papers (P1 and P2) from the F1000RD corpus (Kuznetsov et al., 2022), both dedicated to broad academic topics that are understandable for participants with academic background. Before the study, the participants were instructed about the task, and given 15 minutes to familiarize themselves with the CARE environment. The participants were then split into two groups and assigned paper P1 or P2 based on their group. The participants proceeded to review their assigned paper individually under time constraints (40 minutes), following to the task definition provided above. After the time elapsed, the papers were exchanged between the groups, and the participants were asked to make an acceptance decision for the unseen paper given the inline commentaries produced by a reviewer from the other group. The task was performed in English.

Participants In total 11 researchers from the digital humanities (6) and social sciences (5) participated in the study. A pre-study questionnaire verified that the participant demographics were diverse and that more than 60% of the researchers were at a post-doctoral or professorial level in their careers with adequate English proficiency.

Usability After the study, we conducted a usability survey including a subset of the standardized PSSUQ questionnaire (Borsci et al., 2015), as well as free-form questions (details in Appendix B.2). As Figure 4 shows, the majority of participants were satisfied with using CARE for their task and found that the tool provided adequate speed. Most reported that CARE was clear and easy to use, and

appreciated the sidebar functionality. The survey revealed a few feature requests including the ability to arrange inline commentaries in the sidebar by different criteria, and the ability to leave annotations on figure elements.

Data: Inline commentaries The export functionality allowed us to examine the data resulting from the study. In total, participants created 200 inline commentaries of which 151 were associated with commentary text, 17 ± 7.08 commentaries per user per document on average. The highlight spans comprise of on average 161 ± 151.09 characters and vary vastly from single words up to full paragraphs, selections of two to three sentences being the most common. The associated commentaries have 80 ± 109.98 characters on average, ranging from very short remarks of a single word (e.g. "references?", "why?") to full summarizing paragraphs. These results demonstrate the variability of natural inline commentary use.

Data: Reading behavior Behavioral metrics integrated into CARE allowed us to observe how the participants used the tool to perform the task at hand. We observed that 35 annotations (17.5%) were deleted after creation, prompting us to improve the inline commentary edit functionality in the tool; nearly all participants (70%) made use of the ability to quick-scroll from the in-text highlights to the annotations in the sidebar, while the opposite direction (quick-scroll to the highlight from the sidebar) was only used rarely. The page tracking functionality allowed insights into how participants assessed the papers: by measuring the time spent on each respective page, we established that the participants spent the least amount of time reading bibliography, whereas method and conclusion sections received most scrutiny. We elaborate on these results in the Appendix B.3.

6 CARE and AI Assistance

Data collection CARE enables the collection of inline commentary data that can be used to study inline commentaries and to create new datasets for NLP assistance model development. The collaboration functionality of CARE allows gathering the data about reader interactions within the tool, and the support for free-form tagging and controlled labeling offers great opportunities for collecting user-generated silver data for model pre-training and fine-tuning.

Assisted reading Out of the box, CARE supports integration of any pre-existing *huggingface transformer* model into the reading workflow, which opens a wide range of possibilities for applying previously developed models "in the wild". To provide feedback to the reader, a pre-trained model can be used to enrich inline commentaries with labels, i.e. prompting the reader to provide additional detail, assessing the politeness (Danescu-Niculescu-Mizil et al., 2013), specificity (Li and Nenkova, 2015) or sentiment (Blitzer et al., 2007) of a commentary. In addition, the power and flexibility of modern generative models like T5 (Raffel et al., 2020) allow performing a wide range of text-to-text tasks to assist reading, from question answering to summarization of highlighted passages, with the results rendered as automatically generated replies to the user's inline commentaries. The CARE repository provides sample code for NLP model integration.

Extrinsic evaluation Finally, the behavioral metrics provided by CARE allow to study both how humans read and comment on documents, and how AI assistance impacts this behavior, for example by recording the order in which parts of the document get accessed, or the time needed to create the commentaries. While the current implementation only supports basic time- and location-based measurements, we envision a wide range of extensions that would help us study the impact of AI assistance on reading and text work.

7 Conclusion and Future Work

This paper has presented CARE – a new open platform for the study of inline commentary and AI-assisted reading. CARE enables efficient inline commentary and behavioral data collection for NLP, and supports a wide range of collaborative reading scenarios, while requiring minimal effort to use. The extensible NLP assistance interface allows using CARE for rapid prototyping and extrinsic evaluation of NLP modules that support reading and text-based collaboration. Planned extensions of CARE include support for non-PDF document processing and automatic text highlighting, improved human-in-the-loop functionality and scalability, as well as further development of the onboard behavioral metrics. We invite the community to use our tool and contribute to its further development²⁰.

²⁰<https://github.com/UKPLab/CARE>

Ethics

The experiments performed in this study involved human participants who gave explicit consent to the study participation and to the storage, modification and distribution of the collected data. The arbitrary username selection by the users ensured that the behavioral data did not allow any association with the participants unless they decided to reveal this information. We report the demographic distribution of the participants in the Appendix. The documents used in the study are distributed under an open license. Although we have attempted to reflect the reading-for-peer-review workflow as closely as possible, we note that the study might still not be fully representative of the reading practice during peer review, as the participants were strictly limited in time to perform the task, and the selected papers were not necessarily from the participants' domains of specialist expertise.

Any application of AI to assisting humans in performing real-world tasks bears risk. We stress the need to control for bias, harmful content and factuality of the AI models used to assist reading and text work – especially in the case of large pre-trained generative models. We deem it equally important to educate the users of AI-assisted reading tools about the limitations and risks associated with the integrated assistance models.

From the data collection perspective, we note that all data collected with CARE is human-generated personal data, in particular the behavioral data. We thus *require* the users of the tool to provide explicit informed consent on the data collection upon registration. In addition, the users must explicitly agree with the optional collection of behavioral statistics before any of this data is transferred to the server (opt-in). We stress that while sufficient for controlled studies, in a real application environment these measures would need to be extended by allowing the users to change their decision at a later point and specify the parts of their data that are included into data collection.

From the privacy perspective, CARE allows registration with an arbitrary username, first and last name, e-mail and password. The choice and management of the usernames and user identities are left to the study administrator – we note that if the usernames are not assigned at random and associated with additional data, this needs to be incorporated into the informed consent form upon registration. CARE implements standard security

measures to protect the data, and complete access to the data (documents, inline commentaries, behavioral data) is restricted to the application and server administrator. The security mechanism of the broker and thus of the AI-assistance is currently set via a token defined during the installation of the platform. It is up to the administrator to ensure that the token is kept private, otherwise the models can be used by unwanted users. We stress that for some application scenarios – e.g. dealing with sensitive or confidential documents or performing advanced behavioral measurements – additional security measures should be considered to protect the data.

References

- Tim Baumgärtner, Kexin Wang, Rachneet Sachdeva, Gregor Geigle, Max Eichler, Clifton Poth, Hannah Sterz, Haritz Puerto, Leonardo F. R. Ribeiro, Jonas Pfeiffer, Nils Reimers, Gözde Şahin, and Iryna Gurevych. 2022. [UKP-SQUARE: An online platform for question answering research](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–22, Dublin, Ireland. Association for Computational Linguistics.
- John Blitzer, Mark Dredze, and Fernando Pereira. 2007. [Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification](#). In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.
- Simone Borsci, Stefano Federici, Silvia Bacci, Michela Gnaldi, and Francesco Bartolucci. 2015. Assessing user satisfaction in the era of user experience: Comparison of the sus, umux, and umux-lite as a function of product experience. *International journal of human-computer interaction*, 31(8):484–495.
- 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.
- Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. [A computational approach to politeness with application to social factors](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 250–259, Sofia, Bulgaria. Association for Computational Linguistics.
- Raymond Fok, Andrew Head, Jonathan Bragg, Kyle Lo, Marti A Hearst, and Daniel S Weld. 2022. Scim: In-

- telligent faceted highlights for interactive, multi-pass skimming of scientific papers. *arXiv:2205.04561*.
- Amir Grinstein and Roy Treister. 2017. [The unhappy postdoc: a survey based study](#). *F1000Research*, 6(1642).
- Andrew Head, Kyle Lo, Dongyeop Kang, Raymond Fok, Sam Skjonsberg, Daniel S Weld, and Marti A Hearst. 2021. Augmenting scientific papers with just-in-time, position-sensitive definitions of terms and symbols. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–18.
- Tom Jefferson, Elizabeth Wager, and Frank Davidoff. 2002. Measuring the quality of editorial peer review. *Jama*, 287(21):2786–2790.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. [Survey of hallucination in natural language generation](#). *ACM Comput. Surv.*
- Jan-Christoph Klie, Michael Bugert, Beto Boulosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. [The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation](#). In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.
- Iliia Kuznetsov, Jan Buchmann, Max Eichler, and Iryna Gurevych. 2022. [Revise and Resubmit: An Inter-textual Model of Text-based Collaboration in Peer Review](#). *Computational Linguistics*, 48(4):1–38.
- Junyi Li and Ani Nenkova. 2015. [Fast and accurate prediction of sentence specificity](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1):2281–2287.
- Thomas Morton and Jeremy LaCivita. 2003. [WordFreak: An open tool for linguistic annotation](#). In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Demonstrations*, pages 17–18.
- Mariana Neves and Jurica Ševa. 2019. [An extensive review of tools for manual annotation of documents](#). *Briefings in Bioinformatics*, 22(1):146–163.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). *arXiv:2203.02155*.
- Rik Peels, Rene van Woudenberg, Jeroen de Ridder, and Lex Bouter. 2019. [Academia’s big five: a normative taxonomy for the epistemic responsibilities of universities](#). *F1000Research*, 8(862).
- 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Lorenz Stangier, Ji-Ung Lee, Yuxi Wang, Marvin Müller, Nicholas Frick, Joachim Metternich, and Iryna Gurevych. 2022. [TexPrax: A messaging application for ethical, real-time data collection and annotation](#). *arXiv:2208.07846*.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. [brat: a web-based tool for NLP-assisted text annotation](#). In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. [HuggingFace’s transformers: State-of-the-art natural language processing](#). *arXiv:1910.03771*.

A Application details

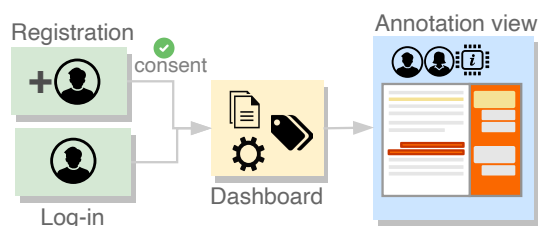


Figure 5: User journey in CARE

User journey Figure 5 illustrates a typical user journey for the reader using CARE. It starts with log-in or registration during which consent and licensing forms are submitted. Afterwards, the user is presented with a dashboard where they can manage documents, label sets and additional settings and export inline commentary and behavioral data. Each document can be opened for reading via the annotation component, where multiple users can annotate the document by leaving inline commentaries organized in a sidebar which also serves as an interface for AI assistance.

Export data format Figure 6 provides an example of the data export functionality: all annotations, comments and discussion threads created by the readers can be directly exported as an easy-to-use JSON. Note that the information presented in the export is also the information available to NLP assistance models to make their predictions.

Behavioral data Figure 7 provides examples of behavioral data that is captured within CARE and exported as JSON objects. Each action is associated with a unique type, meta-data, user information and a timestamp. The captured user interactions include the creation of inline commentary, editing of the same, page scrolling, clicks on important buttons and navigation within the tool.

B User Study Details

This section provides extensive details on the user study setup and results. To recap, the participants were instructed to use CARE for leaving inline commentaries on a manuscript with the purpose of assessing the manuscript’s quality and scientific merit, similar to the critical reading process that takes place during scholarly peer review. Participants were split into two groups that reviewed one manuscript each. Participants subsequently

exchanged the reviewed manuscripts and used the provided inline annotations to decide whether a manuscript should be accepted or rejected, similar to traditional peer review, and surveyed. Figure 8 summarizes the study design. The papers considered were "Academia’s Big Five" (Peels et al., 2019) (P1) and "The Unhappy Postdoc" (Grinstein and Treister, 2017) (P2), both in their first version submitted to F1000 Research.

User Study Context The user study was implemented as a workshop on 25 August 2022 within the Center for Advanced Internet Studies (CAIS)²¹. CAIS is an interdisciplinary research institute in Bochum, Germany, that focuses on the social opportunities and challenges of the digital transformation. Research is conducted in longer-term research programs, as well as by fellows and working groups who are invited to the institute to pursue their own projects. The scientific focus lies on the interface between social sciences, humanities and computer sciences.

B.1 Participant Pool

The participant pool for the user study consisted of 11 CAIS members attending the workshop either virtually (2 participants) or in person (9 participants). No selection criterion was applied to the voluntary participant pool. To ensure the privacy of the participants, we report accumulated frequencies for appropriate value intervals in the following paragraphs.

Demographics Of this participant pool five (45%) identified as women, five (45%) as men and one preferred not to share this information. Around 30% of participants report an age below 40, while the majority of participants lie in the 40 – 49 (45%) age range. The rest of the participants (25%) either lie in the age group above 50 or did not report their age. The majority of participants lived and worked in Germany (80%). We deem the given sample as sufficiently diverse for the purpose of this study, as it covers various age groups and shows nearly balanced genders. However, the age group below forty is under-represented, which might have an influence on the study results, as this particular group might show higher digital affinity. Follow-up studies are required to confirm our findings, where a focus on lower age groups and more diverse nationalities should be considered to account for cultural

²¹<https://www.cais-research.de>

We introduce a new language representation model called **BERT**, which stands for **Bidirectional Encoder Representations from Transformers**. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), **BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers**. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).



Figure 6: Data export example from highlights to annotations in the sidebar, to export JSON.

differences of the partially subjective peer review assessment process.

Academic Background At the moment of the study, more than 60% of the participants were at a post-doctoral or professorial level in their careers, ensuring an adequate level of expertise and experience in scholarly text work. The participants came from diverse academic backgrounds including social studies, philosophy, law, natural language processing and literary studies. The vast majority of participants (90%) had no computer science background.

Reviewing Expertise In an independent pre-study survey among the CAIS members, we confirmed that English language papers and reviews are the predominant form of scientific communication in their respective fields, suggesting adequate language proficiency of the participants during the study.

Roughly 64% of the participants personally reviewed more than one paper in the past year; only two participants reviewed no papers during their career so-far (zero reviews in the past five years). On average the participants reviewed roughly three papers per year. Apart from the prevalent high academic seniority, these numbers generally suggest deep expertise in the task of peer review, while at the same time the study includes participants with little to no reviewing experience.

B.2 Post-study Survey

The participants were asked to fill out the post-study questionnaire directly after the user study. Each participant responded to the web form individually and privately. We ensured the right of erasure under GDPR regulations²² and hosted the questionnaire and resulting data exclusively on EU servers. The questionnaire contained in total 35 items structured into the sections demographics and experience and usability.

Quantitative Results The usability section consists of five general usability questions answered on a seven-point scale ranging from "Strongly disagree" (1) to "Strongly agree" (7), as well as free form questions about missing features and feedback about specific design choices. Figure 4 shows the answer distribution on the usability questionnaire. We asked participants to rate the overall experience using CARE, the speed of usage, the ease of finding information, the comprehensiveness of features and the utility of the sidebar.

Qualitative Results Further on, we asked the participants whether they would prefer different orderings of the comments in the sidebar, where the default during the study was an ordering by text position. While this default is perceived as useful (36%), the option for changing the comment order or other grouping strategies are of interest to the users – especially in the decision making phase based on the inline comments of a reviewer. Subsequently, we asked users to highlight which features

²²<https://gdpr-info.eu/art-17-gdpr>

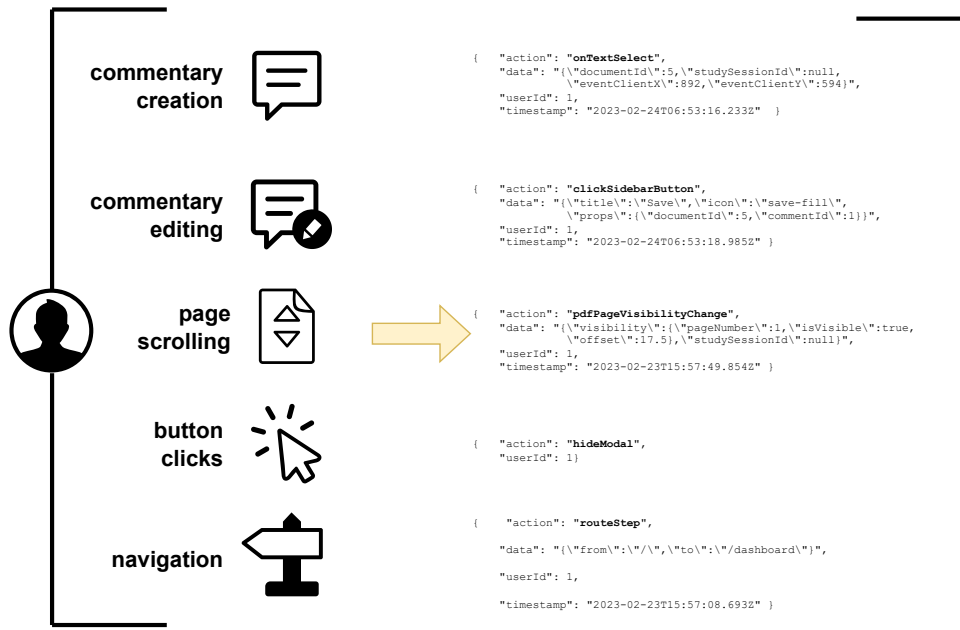


Figure 7: Behavioral user data examples captured and exported as JSON objects.

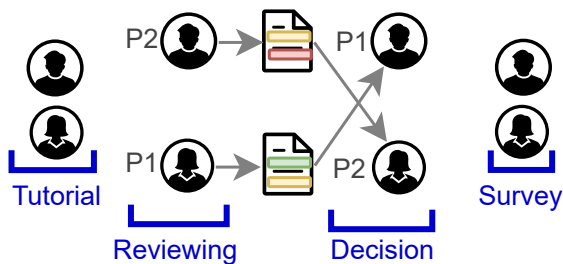


Figure 8: User study setup: Split into two groups after a brief tutorial, the participants review a paper, exchange reviews and make an acceptance decision for the other paper, and participate in the survey.

they missed or could think of to streamline their inline peer review. Most requests were directed towards performing a full peer review based off the inline commentary, e.g. providing notes to editors, providing ratings, having the reviewing guidelines integrated in the interface, etc. Further suggested features that were more focused on the actual highlighting and commenting aspects rather than inline peer review, comprised of more extensive PDF viewer features, like zooming, and improved highlighting features, e.g. sentence-boundary aware highlighting, figure selection, and cross-linking of commentary.

B.3 Behavioral Data

In this section we report on the detailed results of the behavioral data tracking during the user study.

Besides showcasing the behavioral data tracking capabilities of CARE, we intend to collect insights into usage patterns of the tool, as well as establishing a deeper understanding of the use-case of assisting reviewers during reading.

Task Timing We consider several timing metrics to measure the ease of usage, as well as the task difficulty.

First, we measure the time-to-completion, starting with the users accessing the document and ending with them submitting their inline review. The median time-to-completion amounts to 37.82min (just below the provided time limit), with a high standard deviation of roughly 13min. Except for two outliers requiring below 15min, this suggests most people did use and require the full time interval to perform their inline peer review.

Second, we measure the time passed before the interaction with a feature of CARE was registered. This includes text selections for highlights, scrolling to a new page, or creating a comment in the sidebar. We employ this metric as an indicator for the bandwidth of the perceived user interface complexity. In fact, we see that on median 1.28min pass before the first interaction, while again showing high standard deviation of 50s. The high variance and relatively long median time before the first interaction suggest that some participants were still familiarising with the study instructions while already having accessed the document. This shows

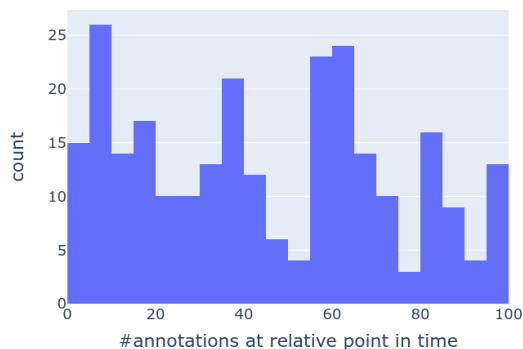


Figure 9: Histogram of the distribution of annotations across time relative to the user’s task timing. We accumulate across users.

one limitation of the behavioral tracking implemented in CARE so-far: while the behavioral data logging is non-obstructive to the user experience, unlike e.g. eye-tracking devices in laboratory scenarios, off-screen activities such as breaks cannot be detected reliably.

Reading and Inline Commentary We consider two metrics to analyze the participants’ focus of attention during the reading process. As the first metric, we consider the time of inline commentary creation relative to the total task time, to quantify whether participants create annotations throughout the reading process or detached before or after reading. For an inline commentary x created at t_c we define the *reltime* relative to the user’s time of entering the document t_e and the time of leaving the document t_l as:

$$reltime(x) = \frac{t_c(x) - t_e}{t_l - t_e}$$

Figure 9 shows the distribution of relative inline commentary timings across participants. Apparently the participants create annotations throughout the whole annotation process, with a light dip at 50%, i.e. after half of the time to completion. These measurements do not suggest that making inline commentary is decoupled from the actual reading process, instead CARE seems to support regular highlighting and note-taking habits while reading.

Turning to the second metric, we compute the time elapsed while viewing a page during the study. We compute the relative reading time per participant and page, considering the two papers in isolation. To estimate the relative time spent per page,

we measure the time deltas between two subsequent page view events, indicating that a PDF page has been rendered on the participants screen, and normalize by the total task time. While this metric is a sufficient approximation for the purpose of assessing the overall reading coverage throughout the document, the measurements on page level instead of scrolling positions limit fine-grained claims about the reading position of a user.

Figure 10 shows the median reading times per page of the users for the two papers in isolation. For both papers, the reading times have a similar "M" shape, where the least amount of time is spent on the very first page, the middle part of the paper and the final pages. For P2 we observe a consistent peak on page two containing the main part of the introduction and, with high variance, page six including the discussion and a central figure of the article. For P1 individual page reading times are less pronounced, but we see peaks on page three (including a large table) and the pages five and six consisting of a long body of text explaining the core contribution (a taxonomy) of the paper.

In the given user study setting, the page viewing times may reveal the parts of the paper that received most scrutiny during reading and commenting, as well as an estimate of the coverage of all paper aspects by the participants. For instance, we see that the bibliography has not been analyzed in detail by any of the participants. In general scenarios, the page viewing times may reveal places of interest in a document or indicate passage that require more effort to process during reading.

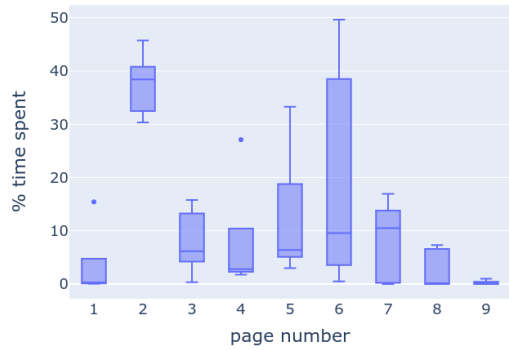
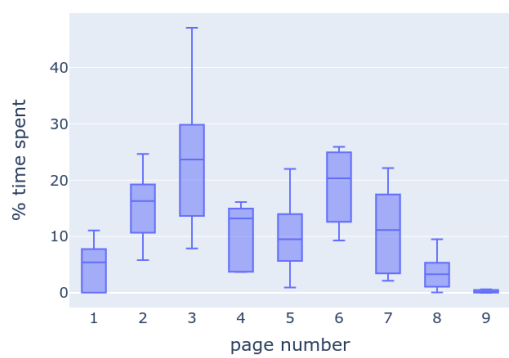


Figure 10: Relative reading time per page for papers P1 (top) and P2 (bottom)

The ROOTS Search Tool: Data Transparency for LLMs

Aleksandra Piktus^{1,2} Christopher Akiki^{3,4} Paulo Villegas⁵ Hugo Laurençon¹
G rard Dupont⁶ Alexandra Sasha Luccioni¹ Yacine Jernite¹ Anna Rogers⁷

¹Hugging Face ²Sapienza University ³Leipzig University ⁴ScaDS.AI
⁵Telefonica I+D ⁶Mavenoid ⁷University of Copenhagen
piktus@huggingface.co

Abstract

ROOTS is a 1.6TB multilingual text corpus developed for the training of BLOOM, currently the largest language model explicitly accompanied by commensurate data governance efforts. In continuation of these efforts, we present the ROOTS Search Tool: a search engine over the entire ROOTS corpus offering both fuzzy and exact search capabilities. ROOTS is the largest corpus to date that can be investigated this way. The ROOTS Search Tool is open-sourced and available on [Hugging Face Spaces](#). We describe our implementation and the possible use cases of our tool.

1 Introduction

Large language models (LLMs) are ubiquitous in modern NLP, used directly to generate text and as building blocks in downstream applications. The ever-increasing size of the latest models inflates the demand for massive volumes of training data (Hoffmann et al., 2022), in practice sourced mainly from the Web. This raises questions concerning the quality of the data, the feasibility of curating and inspecting it, as well as documenting it in terms of what kinds of speech and speakers it represents (Jo and Gebru, 2020; Bender et al., 2021; Akiki et al., 2022). Without that level of characterization, we cannot tell for what varieties of language the resulting models can be expected to work well, whether the data was ethically sourced, how to interpret evaluation metrics, and to what degree a particular output was memorized directly from the training data. In an encouraging new trend, we see researchers exploring ways to quantitatively describe large datasets (Mitchell et al., 2022). However, user-friendly tools for an extensive qualitative analysis are still predominantly missing. In our current work, we aim to fill that gap for a specific, web-scale, textual corpus.

Building on the efforts of the BigScience work-

ROOTS search tool

```
for char, col in zip(string.ascii_letters, counter): vs += vline.VChar(char, (0, col), filename)
logger.debug("char=%s col=%d len=%d vs=%s", char, col, len(vs), str(vs)) def test4(): # rstrip filename
= "/dev/null" vs = vline.VCharString() counter = itertools.count() for char, col in zip("hello, world",
counter): vs += vline.VChar(char, (0, col), filename) logger.debug("char=%s col=%d len=%d vs=%s",
```

Figure 1: ROOTS search tool: user interface

shop,¹ we present the ROOTS Search Tool²—a search engine for the the 1.6TB multilingual ROOTS corpus (Laurençon et al., 2022). The ROOTS corpus was created to pre-train BLOOM (Scao et al., 2022)—the first LLM of its scale designed with commensurate efforts in responsible licensing³ and data governance (Jernite et al., 2022). We hope that our tool will facilitate qualitative analysis of the web-scale ROOTS corpus, and establish the qualitative analysis of training data—critical for the model understanding and governance work—as an essential step in the development of LLMs.

2 Related Work

Corpus linguistics. The core methodology for studying large volumes of text was developed in corpus linguistics (McEnery and Hardie, 2013), an area of research responsible for curating large text collections carefully designed to represent specific varieties of language. For example, the 100M

¹bigscience.huggingface.co

²hf.co/spaces/bigscience-data/roots-search

³bigscience.huggingface.co/blog/the-bigscience-rail-license

word British National Corpus (Leech, 1992) was created to represent the spoken and written British English of the late 20th century, with each text handpicked by experts, who also procured appropriate copyright exemptions. Similar national corpora were later created for many other languages, e.g. Japanese (Maekawa, 2008). The texts were often accompanied by multiple layers of annotations—syntactic, morphological, semantic, genre, source etc. This enabled valuable empirical research on the variants of represented languages, finding use in early distributional semantic models. Corpus linguistics developed sophisticated methodologies including concordances, word sketches and various word association measures (Stefanowitsch and Gries, 2003; Baker, 2004; Kilgarriff, 2014, among others). However, this methodology did not adapt well to Web-scale corpora due to the lack of tools and resources that could support such scale.

Web-scale corpora for LLM pre-training. As LLMs grew, so did the need for massive pre-training datasets. To date, there were several efforts to collect and clean large English and multilingual corpora (Raffel et al., 2020; Xue et al., 2021; Gao et al., 2020; Ortiz Suárez et al., 2020; Bañón et al., 2020; El-Kishky et al., 2020). Non-English, monolingual corpora of this scale have also started to emerge (Gutiérrez-Fandiño et al., 2022; Kummer-vold et al., 2022) However, the sheer scale of such datasets renders them hard to properly curate: we now know that the data used for training LLMs may contain synthetic data (Dodge et al., 2021), privacy-infringing data (Carlini et al., 2020; Huang et al., 2022), incorrect language codes or and translations (Kreutzer et al., 2022), as well as the ubiquitous issues with social biases (Blodgett et al., 2020; Field et al., 2021; Stanczak and Augenstein, 2021, among others). Another issue pertains to the permissions to use the data, which, perhaps the most famously, surfaced in relation to the BookCorpus (Zhu et al., 2015), used, among others, to train BERT (Devlin et al., 2019), but collected without author permissions and eventually taken down by the authors (Bandy and Vincent, 2021).

These issues are a consequence of the fact that the current web-scale corpora are opportunistic samples of publicly available text, rather than artifacts curated to provide a representative snapshot of a specific language variety, as in the corpus linguistics work (Rogers, 2021). This highlights the general problem with the lack of documentation in

NLP datasets of all sizes (Bender and Friedman, 2018; Gebru et al., 2020), and the fact that data work has generally not been a priority in NLP recently (Sambasivan et al., 2021).

Information Retrieval for massive text corpora.

Inspecting large data collection is a central topic of study in another Machine Learning domain, namely Information Retrieval. Even though multiple techniques for analysing large document collections have been developed over the years, there has been little interest so far in applying them specifically to study LLM training data. The closest to our work is that of Dodge et al. (2021) who analyze the C4 dataset (Raffel et al., 2022) and also provide a searchable index.⁴ Similar tools emerge for smaller, more specialised corpora, e.g. COVID-related datasets (Zhang et al., 2020), news quotes (Vuković et al., 2022) and medical literature (Niezni et al., 2022). Razeghi et al. (2022) provide an interface to pre-computed term frequencies from the Pile, but it does not provide full-text corpus search. In the Computer Vision community, related efforts⁵ target large text and image datasets such as LAION (Schuhmann et al., 2022, 2021).

We believe our work to be the first to provide both fuzzy and exact search access to the training corpus of an existing large language model.

3 The ROOTS corpus

The ROOTS corpus (Laurençon et al., 2022) is a high-quality, heterogeneous, multilingual text corpus collected as part of the BigScience project to train the BLOOM LLM (Scao et al., 2022). ROOTS consists of 1.6TB of data in 46 natural and 13 programming languages. The full ROOTS dataset is open to the members of the BigScience Data organization on the Hugging Face hub, which the interested researchers can still apply to join⁶.

3.1 Data Governance

The development of the BLOOM model within the BigScience project was backed by significant work on data governance, as it was identified early on as one of the highest-impact levers of action to enable better accountability and data subject agency in modern ML technology⁷. Participants started by designing a new governance framework to meet

⁴<https://c4-search.apps.allenai.org/>

⁵<https://haveibeentrained.com/>

⁶Sign-up link is available [here](#)

⁷[Data governance and representation in BigScience.](#)

the unique needs of distributed data governance for web-scale data in terms of respecting data subject rights (Jernite et al., 2022). A partial implementation of this framework was used for the ROOTS data as described by Laurençon et al. (2022), focusing on explicit agreements with data custodians, extensive documentation of the data sources, technical tools for privacy-enhancing data handling, and purpose-specific access to subsets of the data.

The present tool goes one step further in implementing the proposed data governance feedback by enabling examination and feedback for the data sources from any interested parties; while still maintaining the controlled access necessary to the proposed governance. The tool only provides 128-word snippets of indexed documents, akin to regular web search engines, and hence provides no practical way to reconstruct the full corpus. The snippets are traceable to their origin in the full ROOTS corpus, and we additionally link to original source documents whenever possible.⁸ Finally, users of the tool are able to flag specific search results and provide an explanation outlining possible infringements of data subjects’ privacy or intellectual property rights. At this stage, the information collected from the flagging process is primarily intended to serve as a basis for future research on collaborative data governance processes. We provide more examples of use cases to support data examination and governance in Section 5.

3.2 Data Pre-processing

Documents vs snippets. ROOTS consists of documents of varying lengths, with outliers as long as 282,571 words. For fuzzy search, we split documents into short snippets of at most 128 words and index snippets rather than the original documents. This helps us follow the controlled access principle discussed in the previous section and makes indexed snippets more comparable in the context of fuzzy search. In exact search, we look for the exact occurrences of the input query within documents and construct snippets ad hoc, including words on both sides of the detected occurrence.

Unique Result IDs. In order to be able to trace search results back to their source, we construct result IDs, adopting the following convention: (a) we include the dataset name as defined on the Hugging Face Hub, followed by (b) the ID of the docu-

⁸The metadata in ROOTS is inconsistent and we only have access to URLs in the pseudocrawl datasets.

Document ID: roots_en_no_code_stackexchange/1495630?seg=para_128_8&seg_id=6


Figure 2: PII leakage: example result for the query gmail.com. We indicate the redacted PII with green and pink treatment.

ment from which the given snippet came, (c) and a question mark. We then include parameters which differ depending on the search strategy used. In fuzzy search we introduce two parameters: the seg parameter describing the segmentation strategy applied during the pre-processing stage, and the seg_id parameter indicating the rank of the given snippet under the specified segmentation strategy. For exact search, we include a single id parameter indicating the the rank of the occurrence of the query in the current document.

PII redaction. During preliminary experiments on the ROOTS corpus, OSCAR (Ortiz Suárez et al., 2019) has been identified as a source of a large amount of documents containing personally identifiable information (PII). A regular-expression-based PII redaction script⁹ has been applied to OSCAR prior to BLOOM training. However, the dataset itself still contains unredacted text. In order to avoid leaking PII through our search tool, we apply an improved variant of the BigScience PII redaction script on the backend side and display results with PII redacted in a visible way - this way one can inspect the data and observe the problem, but personal information are predominantly removed. An example is shown in Figure 2.

4 Implementation

Fuzzy Search Backend. The ROOTS corpus is organized in 498 datasets, each annotated with a language identifier. There are two types of identifiers: those indicating an individual language (e.g. pt for Portuguese), and those indicating a language within a language group (e.g. indic-mr for Marathi, as part of the Indic language group). All programming languages are collected under a common code tag. We build 13 sparse, BM25 (Robertson, 2009) indices: one per language group for the

⁹The BigScience PII redaction script is available [here](#)

ROOTS language tag	# documents	Data size (GB)	# snippets	Index size (GB)	Analyzer
zh, zhs, zht	88,814,841	259.01	111,284,681	682	zh
indic	84,982,982	70.45	100,810,124	714.08	whitespace
en	77,010,827	470.47	695,521,432	766.14	en
es	67,005,817	172.40	267,542,136	264.35	es
fr	58,847,091	204.03	299,938,546	305.29	fr
vi	34,110,375	42.83	76,164,552	72.89	whitespace
pt	31,969,891	77.59	122,221,863	119.98	pt
code	26,176,998	173.16	365,424,222	206.96	whitespace
ar	15,234,080	73.75	68,509,441	93.71	ar
id	12,514,253	19.63	29,531,873	27.16	id
ca	6,142,390	17.42	26,844,600	29.65	es
eu	5,149,797	2.36	6,219,039	4.56	whitespace
nigercongo	1,162,568	0.48	1,462,238	0.89	whitespace
total	597,936,751	1583.59	2,171,474,747	2518.99	

Table 1: Each row represents a single BM25 index we build.

indic and nigercongo groups, one for code, and one for each of the remaining languages (except Chinese, where we combine the tags zh, zht, and zhs into a single index). Table 1 presents the basic information per index. We index respective subsets of the corpus using Pyserini (Lin et al., 2021), a leading toolkit for reproducible IR research. Tokenization is performed with native Lucene¹⁰ analyzers available via Pyserini API (see Table 1 to check which analyzers were used for specific indices).

Exact Search Backend. We leverage a suffix array implementation¹¹ proposed by Lee et al. (2022). We build the suffix array for the whole ROOTS corpus, this time without the split into languages or language groups. We host both the BM25 indices and the suffix array on Hugging Face-provisioned machines. The server code is open-sourced¹².

Frontend and User Experience. The ROOTS Search Tool user interface is built with Gradio (Abid et al., 2019) and served via Hugging Face Spaces.¹³ By default, searches are performed in fuzzy mode, in order to move to the exact search one can enclose the query in double quotes. Fuzzy searches can be performed in a user-specified language, or in *all* languages (in that case results are surfaced separately for each language). We also provide an option to auto-detect the language of the query with a FastText classifier (Joulin et al., 2017). Results are displayed in the order of decreasing relevance; users can control the maximum

number of results they want to see using a slider. In exact search mode, the backend returns all documents matching a given query exactly irrespective of the language, and they are displayed over multiple pages in a random order, with the max results parameter controlling the size of a single page. The total number of matched results is displayed at the top of the results page. PII redaction is applied to all results on the backend side. The tool also allows users to filter out all results from a specific dataset appearing on a given page.

5 Use cases

Detecting PII issues to improve obfuscation. BLOOM was trained with efforts to detect and obfuscate personally identifiable information, or PII (e.g. email and personal addresses, age, phone numbers or government-issued identifiers such as license plates) in the original ROOTS documents. As described in section 3.2, we build on that effort when obfuscating PII in search results. However, it is still possible that some such data was not detected. The tool allows searching for the specific PII by concerned individuals, which is the first step for requesting removal of their data. One could also simply search for their name to see if they are represented in the corpus, and how.

Detecting undesired content. Text from Web crawls contains all kinds of undesired content (Lucioni and Viviano, 2021). Examples of possible classes of problems include hate speech, pornography, synthetic text (e.g. machine-translated text, AI-generated text), word lists that are not meaningful and are meant to trick search engines (Hamilton, 2013), factually incorrect text such as fake news

¹⁰<https://lucene.apache.org/>

¹¹<https://github.com/google-research/deduplicate-text-datasets>

¹²<https://github.com/huggingface/roots-search-tool>

¹³<https://huggingface.co/docs/hub/spaces>

or conspiracy theories, among others. For example, we found at least 5 snippets from the OSCAR source incorrectly arguing that Barack Obama was born in Kenya. While the creators of ROOTS employed filtering strategies targeted specifically at spam and machine-generated content (Laurençon et al., 2022), developing filters for such content is a never-ending arms race with its producers, and the only way to keep improving them is to look at the data—which our tool enables.

Studying representation of dialects and social groups.

When LLM-based systems are deployed, the implicit assumption is often that they are general-purpose and can serve all of its potential users equally well. But there is no such thing as a “neutral”, one-size-fits-all corpus (Rogers, 2021). An obvious issue is dialects, and in case of multilingual models like BLOOM another obvious problem is language imbalance. Besides that, the training data may not equally represent the topics and sources associated with different demographic groups, and hence the LLM would likely not cater to them equally well. Bender et al. (2021) cite the example of GPT-2: the filter for its sources was that they were shared on Reddit, which overrepresents the interests of the typical Reddit user (of whom in the US 67% are men, and 64% are 18-29 years old).

Training data that is then likely to reinforce social stereotypes harmful to marginalized populations. For example, GPT-3 has been shown to over-associate Muslims with violence (Abid et al., 2021). In particular, prompting the model to continue “*Two Muslims walked into...*” tends to lead to mentions of terrorism or assault. BLOOM is not free from these biases: we sampled 10 completions and found 4 that mentioned guns or death (compared to 66% reported for GPT-3). Exact search for “*Two Muslims Walked into...*” returned examples of papers studying this very phenomenon, but a search for just “*Two Muslims*” shows that many passages in OSCAR mention violence or terrorism, whereas mentions in Semantic Scholar, pseudo-crawled websites, and Wikipedia are more varied.

Detecting the presence of specific information.

Where the suitability of a model to a given application depends on it being up-to-date with the latest events, or knowledge about a given fact, a tool like ours can help to quickly find out if the model even theoretically could “learn” a given fact.

For instance, ROOTS contains 231 references to the *death of Queen Elizabeth*, but they refer to the death Elizabeth I in 1603 and not to the recent passing of Elizabeth II in 2022.

Detecting plagiarism/memorization. Generative LLMs can memorize part of their training sets and repeat it verbatim in their outputs. We can probe an LLM to elicit candidates for data memorization (Carlini et al., 2020), and the ROOTS Search Tool can help in different ways:

- By conditioning model probing on actual training data, so that we can more easily check whether such data has been memorized;
- By providing the ground truth to verify that model output was part of the training data;
- By providing the ground truth to verify that model did have a chance to memorize something that it should have memorized;
- By providing match counts to identify which data was more likely to be memorized (since the number of copies in the training data influences memorization (Kandpal et al., 2022)).

For example, BLOOM correctly completes Prince Hamlet’s *To be or not to be* soliloquy—both using greedy decoding and nucleus sampling—but not the less popular Shakespeare quote *I am in this earthly world, where to do harm... is often laudable, to do good sometime accounted dangerous folly*. With our tool we verified that BLOOM had access to at least 7 sources for the *Macbeth* quote (vs at least 47 for *Hamlet*), but did not “learn” it.

Verifying originality. An important question about generative AI models is to what extent their output – that is not a verbatim copy of training data – can be considered original. Consider the above quote from *Macbeth*, which BLOOM completed for us as follows: “*I am in this earthly world, where to do harm... is to do good, and to do good is to do harm.*” With our tool, we could easily verify that the suggested completion does not exist in the corpus verbatim. However, there are dozens of contexts where the concepts of “good” and “harm” are mentioned close to each other (esp. in the phrase “do more harm than good”), so they were the likely indirect sources for this completion. To what degree that completion can be considered new, original text is a key question for the current discussions on plagiarism in AI writing assistants and the legal status of their output.

Non-existing facts. When the same associative mechanism generates factoid text, the model may “hallucinate” events that never occurred—or at least, there was no evidence on which the model could draw. This, too, becomes easy to verify with our tool. BLOOM completed the prompt “*When was the Golden Gate Bridge transported for the second time across Egypt?*” (Hofstadter, 2022) with “*The first time was in the late 19th century, when the bridge was transported from San Francisco to Cairo*”. Of course, this “fact” is untrue, and was not mentioned in the corpus. But we could not even find mentions of anything else transported from San Francisco to Cairo. How exactly LLMs come up with such generations is an interesting research problem, for which tools like ours could be useful.

Enabling data removal requests. The authors of texts that were included in web crawls could use such a tool to identify that fact and request the removal of their texts. For ROOTS, the data governance structure set up for Big Science workshop operated only for its duration, but should there be any future work relying on the same data hosts and agreements, the flagged data collected through our tool can be used to honor the removal requests.

Benchmark data contamination. To interpret benchmark results, we need to know whether they reflect training data memorization or generalization. One approach is for the model authors to specifically plan for the evaluation benchmarks prior to training, and try to exclude the benchmark data (Brown et al., 2020), but this limits the options for external evaluation. Our tool enables sampled checks of benchmark data, and was already successfully used to find¹⁴ that BLOOM should not be evaluated on XNLI (Conneau et al., 2018).

Language contamination. According to Laurençon et al. (2022), ROOTS contains data in 46 languages. But this is clearly not the full story. For example, neither Danish nor Ukrainian are listed, but we found examples in these languages (stackexchange, OSCAR, parsed academic pdf data). The tool can thus be useful for investigating the transfer to “unseen” languages in multilingual evaluation.

Word sense disambiguation. Since the ROOTS Search Tool provides context paragraphs, it can be used to check in what sense a word was used in

¹⁴<https://twitter.com/WilliamBarrHeld/status/1586090252946448384>

the training data. For example, the acronym LLM in ROOTS is used as “large language model” in the parsed academic article data, but in OSCAR it means predominantly “limited liability company” or “Legum Magister”. If future work extends our approach to providing search results through API, then quantitative research would also be possible with techniques like context clustering and classification.

Pre-processing issues. By searching for phrases occurring in different parts of the same document, it is possible to verify that the entire document made it through the pre-processing pipeline – which is useful for improving it. For example, we found a news article in OSCAR, the initial paragraphs of which are missing from ROOTS.

6 Limitations and Future Work

A major limitation of this work is that to mitigate possible issues on the data governance side, we can only provide short snippets of the indexed texts, as is typical of web search engines. We strive to provide links to the original text sources, but this metadata is not consistently available in ROOTS.

Implementation-wise, the current version of exact search is exact down to capitalization and punctuation, and fuzzy search can be noticeably slower. These issues will be addressed in future versions.

The current tool is heavily influenced by the UX of search engines, and its core functionality is similar. In future we intend to review classic corpus analysis tools for ideas of different presentation modes, such as concordance and word sketches. We would like to add more quantitative information, e.g. term frequency information, number of hits, and co-occurrence statistics. Community feedback and suggestions are welcome in the [Community](#) tab of the demo. We are also pursuing a spin-off collaboration with Pyserini to make large scale indexing and hosting of textual data even more seamless.

7 Acknowledgements

We thank the Pyserini team—Odunayo Ogundepo, Xinyu Zhang, Akintunde Oladipo and Jimmy Lin, for their indexing insights. Big thanks to the Gradio team, especially Pete Allen, Abubakar Abid and Freddy Boulton for their support on the front-end side, and to the Hugging Face infra team for answering questions regarding hosting the tool. We thank Carlos Muñoz Ferrandis and Meg Mitchell for valuable discussions.

8 Impact Statement

Our tool aims to improve the current state of documentation search for large corpora of web-scraped text, starting with the ROOTS corpus. However, it also comes with ethical considerations: for instance, it can also inadvertently display sensitive information such as PII and harmful content, and help malicious actors find information about a given topic from multiple sources (which is more difficult given only the raw text of the corpus). We are aware of these limitations, and have taken precautions to compensate for them, such as the PII redaction measures we present in Figure 2. We also present only a snippet of the raw text, which means that for accessing the full documents, users must sign up to be a part of the Big Science organization on the Hugging Face Hub, which also reduces the amount of information that potentially malicious anonymous users can access.

References

- Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*.
- Abubakar Abid, Maheen Farooqi, and James Zou. 2021. [Persistent anti-muslim bias in large language models](#). In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 298–306, New York, NY, USA. Association for Computing Machinery.
- Christopher Akiki, Giada Pistilli, Margot Mieskes, Matthias Gallé, Thomas Wolf, Suzana Ilic, and Yacine Jernite. 2022. [Bigscience: A case study in the social construction of a multilingual large language model](#). In *Workshop on Broadening Research Collaborations 2022*.
- Paul Baker. 2004. [Querying Keywords: Questions of Difference, Frequency, and Sense in Keywords Analysis](#). *Journal of English Linguistics*, 32(4):346–359.
- Jack Bandy and Nicholas Vincent. 2021. [Addressing "documentation debt" in machine learning research: A retrospective datasheet for bookcorpus](#).
- 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 Jaume Zaragoza. 2020. [ParaCrawl: Web-Scale Acquisition of Parallel Corpora](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.
- Emily M. Bender and Batya Friedman. 2018. [Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science](#). *Transactions of the Association for Computational Linguistics*, 6:587–604.
- 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 *FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.
- Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. [Language \(Technology\) is Power: A Critical Survey of "Bias" in NLP](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, 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, 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.
- Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. [Extracting Training Data from Large Language Models](#). *arXiv:2012.07805 [cs]*.
- Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. [XNLI: Evaluating cross-lingual sentence representations](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. 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.
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. [Documenting](#)

- Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2020. **CCAligned: A Massive Collection of Cross-Lingual Web-Document Pairs**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5960–5969, Online. Association for Computational Linguistics.
- Anjalie Field, Su Lin Blodgett, Zeerak Waseem, and Yulia Tsvetkov. 2021. **A Survey of Race, Racism, and Anti-Racism in NLP**. 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 1905–1925, Online. Association for Computational Linguistics.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. **The Pile: An 800GB Dataset of Diverse Text for Language Modeling**. *arXiv:2101.00027 [cs]*.
- Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2020. **Datasheets for Datasets**. *arXiv:1803.09010 [cs]*.
- Asier Gutiérrez-Fandiño, David Pérez-Fernández, Jordi Armengol-Estapé, David Griol, and Zoraida Callejas. 2022. **esCorpius: A Massive Spanish Crawling Corpus**.
- Peter A. Hamilton. 2013. Google-bombing - manipulating the pagerank algorithm. *Information Retrieval*.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. **Training compute-optimal large language models**.
- Douglas Hofstadter. 2022. **Artificial neural networks today are not conscious, according to Douglas Hofstadter**. *The Economist*.
- Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. **Are Large Pre-Trained Language Models Leaking Your Personal Information?**
- Yacine Jernite, Huu Nguyen, Stella Biderman, Anna Rogers, Maraim Masoud, Valentin Danchev, Samson Tan, Alexandra Sasha Luccioni, Nishant Subramani, Isaac Johnson, Gerard Dupont, Jesse Dodge, Kyle Lo, Zeerak Talat, Dragomir Radev, Aaron Gokaslan, So-maieh Nikpoor, Peter Henderson, Rishi Bommasani, and Margaret Mitchell. 2022. **Data governance in the age of large-scale data-driven language technology**. In *2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22*, page 2206–2222, New York, NY, USA. Association for Computing Machinery.
- Eun Seo Jo and Timnit Gebru. 2020. **Lessons from archives**. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM.
- 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.
- Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. **Deduplicating Training Data Mitigates Privacy Risks in Language Models**. In *Proceedings of the 39th International Conference on Machine Learning*, pages 10697–10707. PMLR.
- Adam Kilgarriff. 2014. **The Sketch Engine: Ten years on**. *Lexicography*, pages 1–30.
- Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iro-ro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhlov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. **Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets**. *Transactions of the Association for Computational Linguistics*, 10:50–72.
- Per Kummervold, Freddy Wetjen, and Javier de la Rosa. 2022. **The Norwegian Colossal Corpus: A Text Corpus for Training Large Norwegian Language Models**. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3852–3860, Marseille, France. European Language Resources Association.
- Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg

- Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Romero Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Vu Minh Chien, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Luccioni, and Yacine Jernite. 2022. [The bigscience ROOTS corpus: A 1.6TB composite multilingual dataset](#). In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. [Deduplicating training data makes language models better](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
- Geoffrey Neil Leech. 1992. 100 million words of English: The British National Corpus (BNC). *Language Research*, 1/4.
- Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. [Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations](#). In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.
- Alexandra Luccioni and Joseph Viviano. 2021. [What’s in the box? an analysis of undesirable content in the Common Crawl corpus](#). 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 182–189, Online. Association for Computational Linguistics.
- Kikuo Maekawa. 2008. [Balanced Corpus of Contemporary Written Japanese](#). In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 101–102.
- Tony McEnery and Andrew Hardie. 2013. [The History of Corpus Linguistics](#). In *The Oxford Handbook of the History of Linguistics*. Oxford University Press.
- Margaret Mitchell, Alexandra Sasha Luccioni, Nathan Lambert, Marissa Gerchick, Angelina McMillan-Major, Ezinwanne Ozoani, Nazneen Rajani, Tristan Thrush, Yacine Jernite, and Douwe Kiela. 2022. [Measuring data](#). *CoRR*, abs/2212.05129.
- Danna Niezni, Hillel Taub-Tabib, Yuval Harris, Hagit Sason-Bauer, Yakir Amrusi, Dana Azagury, Maytal Avrashami, Shaked Launer-Wachs, Jon Borchardt, M Kusold, Aryeh Tiktinsky, Tom Hope, Yoav Goldberg, and Yosi Shamay. 2022. [Extending the boundaries of cancer therapeutic complexity with literature data mining](#). *bioRxiv*.
- Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. [A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.
- Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. [Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures](#). In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim. Leibniz-Institut für Deutsche Sprache.
- 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.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *The Journal of Machine Learning Research*, 21(1):140:5485–140:5551.
- Yasaman Razeghi, Raja Sekhar Reddy Mekala, Robert L Logan Iv, Matt Gardner, and Sameer Singh. 2022. [Snoopy: An Online Interface for Exploring the Effect of Pretraining Term Frequencies on Few-Shot LM Performance](#). In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 389–395, Abu Dhabi, UAE. Association for Computational Linguistics.
- S. Robertson. 2009. [The Probabilistic Relevance Framework: BM25 and Beyond](#). *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Anna Rogers. 2021. [Changing the World by Changing the 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 2182–2194, Online. Association for Computational Linguistics.
- Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. ["Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI](#). In *Proceedings of the 2021 CHI Conference on Human*

Factors in Computing Systems, CHI '21, pages 1–15, New York, NY, USA. Association for Computing Machinery.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Amanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nuru-laqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, So-maieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M. Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper,

Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, San-chit Gandhi, Shaden Smith, Stéphane Reuena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névél, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Undreaaj, Arash Aghagol, Are-zoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Onon-iwu, Habib Rezanejad, Hessie Jones, Indrani Bhat-tacharya, Irene Solaiman, Irina Sedenko, Isar Ne-jadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim El-badri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Ra-jani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Al-izadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguié, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Perriñán, Daniel Molano, Dian Yu, Enrique Manjavac-cas, Fabio Barth, Florian Fuhrmann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangas-sai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A. Castillo, Mari-anna Nezhurina, Mario Sängler, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myung-sun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda,

- Shlok S. Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aaronsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022. [BLOOM: A 176B-Parameter Open-Access Multilingual Language Model](#). In *Thirty-Sixth Conference on Neural Information Processing Systems*, New Orleans, Louisiana. arXiv.
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W. Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R. Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. [LAION-5B: An open large-scale dataset for training next generation image-text models](#). In *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. [LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs](#). In *Data Centric AI NeurIPS Workshop 2021*.
- Karolina Stanczak and Isabelle Augenstein. 2021. [A Survey on Gender Bias in Natural Language Processing](#).
- Anatol Stefanowitsch and Stefan Th Gries. 2003. [Collostructions: Investigating the interaction of words and constructions](#). *International Journal of Corpus Linguistics*, 8(2):209–243.
- Vuk Vuković, Akhil Arora, Huan-Cheng Chang, Andreas Spitz, and Robert West. 2022. [Quote erat demonstrandum: A web interface for exploring the quotebank corpus](#). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.
- 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.
- Edwin Zhang, Nikhil Gupta, Raphael Tang, Xiao Han, Ronak Pradeep, Kuang Lu, Yue Zhang, Rodrigo Nogueira, Kyunghyun Cho, Hui Fang, and Jimmy Lin. 2020. [Covidex: Neural ranking models and keyword search infrastructure for the COVID-19 open research dataset](#). In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 31–41, Online. Association for Computational Linguistics.
- Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. [Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books](#). In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27.

The OPUS-MT Dashboard – A Toolkit for a Systematic Evaluation of Open Machine Translation Models

Jörg Tiedemann and Ona de Gibert
Department of Digital Humanities
University of Helsinki, Helsinki / Finland
{jorg.tiedemann, ona.degibert}@helsinki.fi

Abstract

The OPUS-MT dashboard is a web-based platform that provides a comprehensive overview of open translation models. We focus on a systematic collection of benchmark results with verifiable translation performance and large coverage in terms of languages and domains. We provide results for in-house OPUS-MT and Tatoeba models as well as external models from the Huggingface repository and user-contributed translations. The functionalities of the evaluation tool include summaries of benchmarks for over 2,300 models covering 4,560 language directions and 294 languages, as well as the inspection of predicted translations against their human reference. We focus on centralization, reproducibility and coverage of MT evaluation combined with scalability. The dashboard can be accessed live at <https://opus.nlp1.eu/dashboard/>.

1 Introduction

The main motivation behind the OPUS-MT dashboard is to provide a comprehensive overview of open translation models. We focus on a systematic collection of benchmark results with verifiable translation performance and large coverage in terms of languages and domains. The landscape of Machine Translation (MT) is increasingly blurry and incomprehensible due to the growing volume of shared tasks and models published within the community. Even with established events such as the Conference on Machine Translation (WMT), a complete picture of translation performance is hard to obtain. In addition, large multilingual language and translation models push the language coverage making it difficult to keep an eye on the state of the art for particular language pairs.

One additional problem is that most results reported in scientific and non-scientific channels come from selected benchmarks and model performance and are not explicitly verified by a careful replication study. In various cases, new models

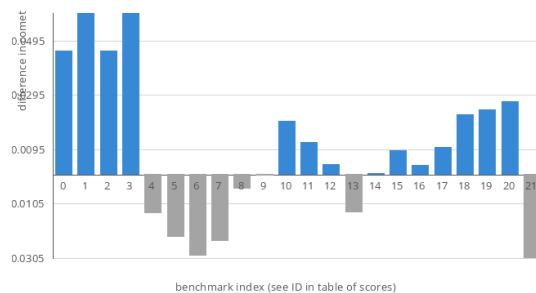


Figure 1: Difference between two models in terms of COMET scores across 21 benchmarks for English-to-French translation.

come with their own benchmarks and do not consider a wider evaluation across domains. Training data is complicated to control and the danger of over-fitting to specific scenarios is apparent. Figure 1 illustrates the substantial differences one can observe across benchmarks when comparing two competing models.

Our dashboard is an attempt to carefully provide a summary of results using an extendable collection of benchmarks with the largest language coverage possible accommodated with procedures to translate and evaluate in a standardized and consistent setup. The focus is clearly set on publicly available translation models as we want to emphasize translation results that we can replicate and verify from our own experience. The system is designed with the following requirements in mind:

- comprehensive and scalable collection
- lean and responsive interface
- open and transparent implementation

The implementation and all data files are available on GitHub in public repositories and details about the components and implementations are given below. We start by a brief description of the background before discussing the collection of benchmarks and translation evaluations. The main features of the web interface are listed thereafter

and we finish up with links to related work and an outlook into future developments.

2 Background and Motivation

The main motivation of the dashboard is related to our own initiative on building open translation models under the umbrella of OPUS-MT. The development of MT accelerated in recent years making it difficult to obtain a clear view on performance for individual language pairs. In contrast to related work, the dashboard is not intended to serve as a new MT evaluation service but rather as a central point of comparison between OPUS-MT and other publicly available models. User-provided translations are also supported as another point of reference but the focus is set on verifiable translation results produced by the system itself.

OPUS-MT is based on OPUS (Tiedemann, 2012), the major hub of public parallel data, the main ingredient for training modern translation models. It refers to an initiative to systematically train translation models on open data sets using the efficient NMT framework provided by Marian-NMT (Junczys-Dowmunt et al., 2018). The project includes both bilingual and multilingual transformer models of different sizes with streamlined procedures to scale up language coverage and availability (Tiedemann and Thottingal, 2020). OPUS-MT models are released with permissive licenses and can easily be integrated in existing workflows (Tiedemann et al., 2022; Nieminen, 2021; Tiedemann et al., 2023). Currently, there are over 2,300 public models – release information is part of the dashboard.¹ The models cover over 290 languages and provide translations for 2,549 language pairs and 4,560 language directions. The sheer volume of OPUS-MT releases call for a systematic evaluation to monitor progress and support model selection in practical applications.

3 Collecting MT Benchmarks

MT benchmarks are developed for various tasks and domains and their distribution differs depending on the preferences of the original provider. In order to make it easier to systematically compare MT across available benchmarks, we collect known testsets in a unified and consistent format in a public repository² (OPUS-MT-testsets).

¹<https://opus.nlpl.eu/dashboard/releases.php>

²<https://github.com/Helsinki-NLP/OPUS-MT-testsets/>

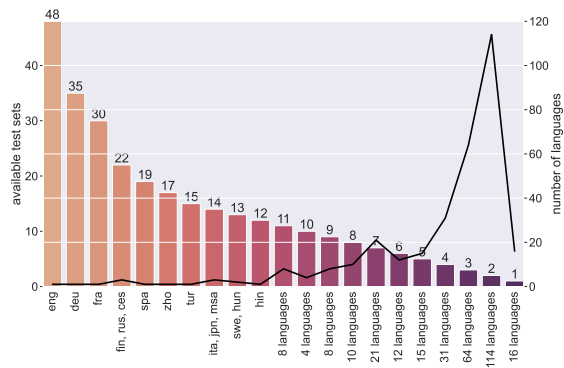


Figure 2: Distribution of languages covered in public benchmarks bucketed by the number of available test sets per language. The solid line shows the number of languages in each bucket.

The current collection covers benchmarks from translation tasks at WMT (Kocmi et al., 2022), TICO19 (Anastasopoulos et al., 2020), the Tatoeba translation challenge (Tiedemann, 2020), Flores v1.0 (Guzmán et al., 2019), Flores-101 (Goyal et al., 2022) and Flores-200 (NLLB Team et al., 2022a) and multi30k (Elliott et al., 2016). Each benchmark may be comprised of several testsets (different years or dev and test sets). Altogether we cover over 44,000 language directions with an average length of 1,945 sentences per testset. One important thing to note is that the multilingual benchmarks are typically English-centric in a sense that the original text has been translated from English to other languages. Figure 2 illustrates the skewed distribution and we encourage suggestions³ and developments of additional sources to change that picture.

We sort benchmarks by language pair using ISO-639-3 language codes and use a simple plain text format with UTF-8 encoded data. Translated segments (typically sentences) appear on the same line in separate files for the input text in the source language and reference translations in the target language. The file name corresponds to the benchmark name and the file extension specifies the language by its ISO code. Additional files may provide additional metadata such as domain labels or source information. We also add extensions about the writing script if necessary. Here are a few examples from the collection:

```
eng-deu/newstest2020.deu
eng-deu/newstest2020.eng
```

³Suggestions can be proposed by adding issues to our GitHub repository.


```
srp_Cyrl-fin/flores200-devtest.fin
srp_Cyrl-fin/flores200-devtest.srp_Cyrl
```

Currently, we do not support multi-reference benchmarks. Instead, additional reference translations are stored as separate benchmarks. Document boundaries may be specified by empty lines. Other formatting information is not supported.

We will extend the repository with additional test sets including NTREX (Federmann et al., 2022), Samanantar (Ramesh et al., 2021), IWSLT (Antonios et al., 2022) and data from the Masakhane project (Orife et al., 2020).

4 Systematic Evaluation of Public Models

We store the results of our systematic evaluation in three different public git repositories, depending on the type of model: (i) Opus-MT models⁴, (ii) external models⁵, and (iii) user-contributed translations⁶.

We emphasize a lean design avoiding the hassles of setting up and maintaining databases and additional services. Each leaderboard repository follows the same structure and is divided into two main parts: (i) leaderboards for each benchmark and language pair and (ii) the scores for each individual model. File structures are organized accordingly and the setup makes it possible to easily scale the collection to a large number of models, benchmarks and language pairs. The inclusion of new evaluation benchmarks is also straightforward as we have separate files for each of them. The main file structure looks like this:

```
scores/<src>-<trg>/<test>/<metric>-scores.txt
models/<org>/<model>/<test>.<metric>-scores.txt
```

Source and target language IDs (<src>, <trg>) and the name of the benchmark (<test>) correspond to the naming conventions used in OPUS-MT-testsets. Supported metrics are currently COMET (Rei et al., 2020),⁷ BLEU (Papineni et al., 2002) and chrF (Popović, 2015) with the two variants chrF++ (Popović, 2017) and sentence-piece-based subword BLEU (Goyal et al., 2022). We use sacrebleu (Post, 2018) and unbabel-comet⁸ to compute the results. We add the readily available

⁴<https://github.com/Helsinki-NLP/OPUS-MT-leaderboard/>

⁵<https://github.com/Helsinki-NLP/External-MT-leaderboard/>

⁶<https://github.com/Helsinki-NLP/Contributed-MT-leaderboard/>

⁷COMET model: Unbabel/wmt20-comet-da

⁸<https://pypi.org/project/unbabel-comet>

Chinese, Japanese and Korean tokenization features in sacrebleu and the Flores-200 sentence piece model for subword splitting. Models are sorted by provider (<org>); and the model name (<model>) may be split into sub-directories specifying additional properties of the model. For example, OPUS-MT models are grouped by languages they support, which may be a single language pair or pairs of language groups.

Besides benchmark-specific score tables, we also compile aggregated score tables for each language pair. Those tables list an average score over several available benchmarks (avg-<metric>-scores.txt) and the best-performing model for each available benchmark (top-<metric>-scores.txt). Automatic makefile recipes are used to update those tables if needed. We keep separate tables for the three categories (OPUS-MT, external models, user-contributed translations) in each of the respective repositories.

Furthermore, the repository includes the procedures for translating and evaluating models with respect to the benchmarks collected as described in the previous section. The translations are systematically run with the same hyperparameters. These can be consulted in Appendix A. The implementation streamlines the creation of batch jobs and enables a scalable approach that allows a straightforward update of leaderboards with new models and benchmarks. Additional evaluation metrics can also be integrated by implementing appropriate recipes.

The general workflow for evaluating models and updating score tables is divided into three steps: (i) translating and evaluating all benchmarks for all language pairs supported by a model, (ii) registering new scores to be added to existing leaderboards, and (iii) updating all affected leaderboards and sorting by score. GNU makefile recipes are used to properly handle dependencies. Using revision control as the backend storage makes it possible to recover from errors and mistakes.

We distinguish between internal and external models but the basic workflow is the same. We anticipate that the publication of this paper will attract some interest allowing us to harvest additional user-contributed translations. We report statistics in Table 1. The relatively small amount of external models and the high number of translations is due to the fact that some of the external models

	OPUS-MT	Tatoeba	External
Providers	-	-	34
Models	693	1,633	78
Bilingual	634	779	73
Multilingual	59	854	5
Lang pairs/model	1.93	11.00	36.42
Translations	5,483	80,295	11,723
Testsets	49	66	49
Language directions	650	4,306	786
Language pairs	372	2,388	504
Language coverage	99	276	219

Table 1: Statistics on OPUS-MT, Tatoeba and external models.

are highly multilingual, in comparison, OPUS-MT models are mostly bilingual. Some additional details are given below.

4.1 OPUS-MT Models

OPUS-MT models are released as self-contained Marian-NMT models. They come in two flavors: Models trained on different selections from OPUS (Tiedemann and Thottingal, 2020)⁹ and models trained on OPUS data released as part of the Tatoeba translation challenge (Tiedemann, 2020).¹⁰ Release information is available from GitHub and feeds directly into the evaluation workflow. An update to the collection triggers new evaluation jobs. Missing benchmark scores can also be added using the dependency chains implemented in the leaderboard workflow. The actual jobs still need to be started manually as we have to control compute time allocations on the infrastructure that we employ. Translations and evaluations are typically done on some high-performance cluster (scheduled using SLURM (Yoo et al., 2003)) and we integrated the OPUS-MT leaderboard in the environment provided by CSC, the center for scientific computing in Finland.¹¹ However, all jobs should execute in any environment where all prerequisites are properly installed (mainly MarianNMT, sacrebleu, comet-scorer, and SentencePiece).

4.2 External Models

Comparing OPUS-MT to other public models is important to monitor their performance in relation to the state of the art. The extension of the OPUS-MT dashboard to external models is currently supported

⁹<https://github.com/Helsinki-NLP/Opus-MT/>

¹⁰<https://github.com/Helsinki-NLP/Tatoeba-Challenge/>

¹¹<https://www.csc.fi/>

		Models	%
(1)	Text2TextGeneration/Translation	11,662	100.00
(2)	Helsinki-NLP	1,439	12.34
	Potential candidates	10,223	87.66
(3)	With no language metadata	7,643	65.54
	With only one language tag	2,166	18.57
	With at least two language tags	414	3.55
(4)	Identifiable language direction	144	1.23

Table 2: Statistics of available models on the Huggingface repository with metadata on the targeted tasks.

by the model hub from Huggingface.¹² Metadata tags allow us to search the hub efficiently by filtering by task and language.

We proceed as follows. (1) First, we search the hub to select models that are tagged for tasks *Translation* or *Text2TextGeneration*. (2) Then, we discard Helsinki-NLP models which are already included in the dashboard and (3) keep those models that have at least two language tags. Since the platform does not provide source and target tags, (4) we try to infer the language direction from the model’s name by using regular expressions, a naive but effective solution. (5) Finally, we only keep the models that can be used with the translation pipeline straight out of the box. During this process, we encountered several issues such as the need for non-standard language parameters (e.g en_XX instead of en) or the need for batch size optimization.

We report statistics on the models encountered at each of the steps in Table 2. Surprisingly, only 3.55% of the models have at least two language tags and thus, are potential candidates for our purpose. Although we acknowledge that this is a small subset when compared to the extensive range of available models, the scalability of the method is granted as long as there is sufficient metadata. The lack of documentation on the hub is an issue far beyond our reach. However, we advocate for developers to document models to the fullest extent possible.

Furthermore, apart from the models obtained with the process mentioned above, we specially target large multilingual models to cover as many languages as possible. We added the three following models in various sizes: M2M-100 (Fan et al., 2020), No Language Left Behind (NLLB) (NLLB Team et al., 2022b) and MBART (Tang et al., 2020).

¹²<https://huggingface.co/models>

4.3 User-Contributed Translations

In addition to incorporating internal and external models with our own evaluations, the dashboard also provides an opportunity for the community to contribute their collected translations. Translations for a specific benchmark can be easily added following a makefile recipe.

In this context, we have envisioned mainly two scenarios. This feature is highly beneficial to report results for very large MT models that entail high computational costs, such as NLLB’s largest variant with 54.5 B parameters. We have collected its scores from <https://tinyurl.com/nllbflorestranslations> and they can be consulted from the dashboard. Secondly, this feature offers flexibility and is especially suitable when researchers aim to include their own models in the dashboard, even if we do not internally run those models. In both cases, scores of user-contributed translations are displayed separately, as their reliability is lower since we did not produce the translations ourselves.

5 MT Performance Dashboard

For the dashboard web-interface, we emphasize a lightweight implementation. In our system, we want to avoid a complex backend and heavy frontends and rather focus on lean and responsive functionalities. The interface has minimal requirements and basically runs with a standard PHP-enabled web server. Data is automatically fetched from the OPUS-MT storage, GitHub or the local file system. Deployment requires no further installation or database setups. The frontend uses cookies and session variables to speed up the process but can also run without them. Server-side caching is used to enable fast response time and no heavy graphics or animations are used that would slow down data transfer and client-side website rendering.

5.1 Benchmark Summaries

The main functionality of the dashboard is to provide summaries of translation performance coming from automatic evaluation. It automatically connects with the relevant repositories described above making their content immediately visible in the interface. Three basic benchmark views are implemented: (i) A summary over best-performing models for a selected language pair over all available benchmarks, (ii) an overview of translation models evaluated on a specific benchmark, and (iii)

	afr	dan	deu	eng	fao	isl	ltz	nld	nno	nob	swe
afr		26.4	20.1	43.8	5.1	13.9	5.6	17.3	19.0	20.2	25.3
dan	23.2		24.1	37.8	6.0	14.5	6.2	18.5	22.5	24.5	32.4
deu	20.7	26.9		32.0	4.5	12.8	7.8	18.5	17.2	18.5	25.4
eng	32.4	36.6	26.6		6.1	16.0	6.3	20.2	23.1	25.7	35.3
fao	10.0	13.8	9.5	14.1		8.1	3.3	8.3	10.3	10.1	12.5
isl	15.5	18.7	14.6	23.2	5.0		4.1	12.0	13.9	14.9	17.8
ltz	14.4	17.2	18.6	21.2	3.4	8.6		11.5	11.5	12.9	15.7
nld	16.5	20.0	17.6	24.1	4.5	9.6	4.9		13.1	15.0	19.2
nno	21.5	30.0	20.8	34.4	5.7	13.9	5.4	16.5		24.6	28.7
nob	20.3	26.2	18.3	33.7	5.3	12.3	4.9	16.4	20.4		25.7
swe	23.3	33.3	23.8	37.8	5.6	14.3	6.1	18.5	22.1	23.8	

Figure 3: A heatmap of BLEU scores for a multilingual OPUS-MT model covering Germanic languages. Target languages refer to columns in the table.

a comparison of two selected models on available benchmarks (see Figure 1). All views come with simple bar charts and tables linked to relevant information and downloads (see screenshots in the appendix). For multilingual translation models we also added a matrix view in terms of a heatmap illustrating scores across all evaluated language pairs (see Figure 3).

In all modes, the evaluation metric can be selected and other views are linked to quickly jump between them. We provide download links for internal models and links to models’ websites when available for metadata regarding each model’s characteristics. All system translations and evaluation log files are also available for download to make the process as transparent as possible.

5.2 Inspecting Translations

Another important feature is the possibility to browse through the actual translations produced by each model. We provide all of the translations together with reference translations from the original benchmark in order to study the differences between proposed and human translations. In addition, it is also possible to compare the output of two models on the same benchmark. Highlighting differences can be enabled using a word-level diff function (see Figure 4). The interface displays 10 translation instances at a time but the whole dataset can be downloaded and inspected offline.

6 Related Work

In the current scenario of NLP, which is characterized by the increasing number of available models versus the constant lack of systematic documentation, von Werra et al. (2022) identify three challenges: (1) *reproducibility*, the replicability of model performance, (2) *centralization* to avoid the

SOURCE: You can't do it on your own.
REFERENCE: Allein schaffst du das nicht.
MODEL 1: Du kannst es nicht alleine machen.
MODEL 2: Du kannst es nicht alleine schaffen.
DIFF: Du kannst es nicht alleine [-machen.-] {+schaffen.+}

SOURCE: All the people were moved by his speech.
REFERENCE: Alle Leute waren von seiner Rede gerührt.
MODEL 1: Das ganze Volk war von seiner Rede bewegt.
MODEL 2: Alle Menschen wurden von seiner Rede bewegt.
DIFF: [-Das ganze Volk war-] {+Alle Menschen wurden+} von seiner Rede bewegt.

Figure 4: Browsing through translation examples, comparing the output of two models and reference translations for a specific benchmark (Tatoeba English-German).

duplication of workload and one single point of reference, and (3) *coverage*, the inclusion of a diverse set of metrics and languages, as well as pointers for efficiency, bias and robustness.

One of the most well-known efforts on reproducibility is Papers With Code¹³ (PWC), an open-source web-based platform that links papers with their implementations and includes many features, such as benchmark evaluations and information on the use of dataset. The Huggingface repository also aims at solving the issue of reproducibility with their Evaluation on the Hub (von Werra et al., 2022), a user-friendly platform that allows large-scale evaluation of any of their publicly available models. This feature holds significant potential, however, two primary concerns arise. Firstly, the evaluation is not done systematically, as assessments are performed solely upon user request. Secondly, although the openness of the hub to all users is a move towards democratizing ML, it also results in an absence of metadata for the uploaded resources that make it difficult to find suitable models for every dataset. Yet another similar tool is Dynabench (Kiela et al., 2021), a research platform for dataset creation and dynamic model benchmarking for a wide range of NLP tasks.

Generic platforms mentioned above are useful but make it difficult to get a comprehensive overview of one specific task. MT has a long tradition of shared tasks and several systems have been developed to visualize and analyze benchmark results. The WMT matrix¹⁴ was a dedicated platform for submitting and archiving submissions to the translation tasks at WMT. It provided a useful overview of the state-of-the-art in those tasks, but the system is down.

MT-CompareEval (Klejšch et al., 2015) partially

replaces that service with its WMT instance.¹⁵ The system itself provides a modern tool for the analysis of MT output with various plots of automatic evaluation metrics and an interface for comparing translations with highlighted differences. The software is open source and can be deployed with the option to upload additional system translations, which is also used by the developers to host their own experimental results.¹⁶ In contrast to the OPUS-MT dashboard, it does not implement the replication of translations to verify provided results.

Another open-source tool, compare-mt (Neubig et al., 2019), has been developed to explore translation outputs. It supports a deeper analysis and comparison using detailed statistics of word-level and sentence-level accuracies, salient n-grams, etc.

On the commercial side, there is Intento, a language service provider that publishes a yearly report with an overview of current MT systems Savenkov and Lopez (2022) with a focus on commercial models. They provide an end-to-end MT evaluation service that comes with a cost and the yearly evaluation is not transparent and open.

7 Conclusions and Future Work

The OPUS-MT dashboard implements a simple yet comprehensive interface for a systematic evaluation of public translation models. The main purpose is to provide an overview of OPUS-MT models and to relate their performance to other openly available models. The focus is on verifiable performance and a centralized evaluation procedure. The workflow and collection stress transparency and replicability and can easily be extended with new models and benchmarks.

The current implementation is fully functional

¹³<https://paperswithcode.com>

¹⁴<http://matrix.statmt.org>

¹⁵<http://wmt.ufal.cz/>

¹⁶<http://mt-compareval.ufal.cz/>

Provider	Model	Parameter Size	Batch size	Benchmarks / hour
facebook	m2m100_418M	418M	16	17.47
facebook	m2m100_418M	418M	16	14.14
facebook	m2m100_418M	418M	16	15.25
facebook	m2m100_1.2B	1.2B	8	8.97
facebook	m2m100_1.2B	1.2B	8	8.14
facebook	m2m100_1.2B	1.2B	8	10.47
facebook	nllb3.3B	3.3B	16	11.03
facebook	nllb3.3B	3.3B	16	11.78
facebook	nllb3.3B	3.3B	16	12.03
facebook	nllb3.3B	3.3B	2	4.18
facebook	nllb3.3B	3.3B	2	3.94
facebook	nllb3.3B	3.3B	2	3.19

Table 3: Approximate inference statistics (number of translated benchmarks per GPU hour) on three large multilingual models with different sizes and different runs on a single NVIDIA Ampere A100 GPU.

but we already work on several extensions. First of all, we would like to integrate more information about the model properties in the dashboard. Important features are model size, inference time and computational costs that can be related to translation performance. Additionally, we want to tag other important characteristics such as multilingual versus bilingual models. Heatmaps for comparing multilingual model scores are also on our to-do list as well as better overviews of top-scores in multilingual benchmarks. We also want to integrate our geolocated visualization of language coverage implemented in OPUS-MT map.¹⁷

Finally, we are continuously working on the integration of new benchmarks and the systematic evaluation of available models. We look into other released models and their use for replicating benchmark results. We also continue to collect benchmarks and will integrate sentence-level scores while browsing through translation output. We may also connect to other systems like MT-CompareEval for more detailed analyses.

Limitations

In this paper, we have introduced OPUS-MT dashboard, our system for MT evaluation with a focus on centralization and reproducibility. One of the limitations of the presented approach is that the current coverage is based solely on automatic MT metrics. Nevertheless, as mentioned above, we are working towards adding pointers for model size, inference time and computational costs. A large scale manual assessment is beyond our capabilities. However, we consider the option to enable community-driven feedback that could help to add

human judgments to the system outputs. Furthermore, at the moment we are not aware of how we can add information regarding bias and fairness, but we will look into additional points of information that can be added to the collection.

Another limitation of our method is that for the multilingual external models, currently we only provide English-centric translations (en-xx, xx-en), due to the high computational costs of running inference on large language and translation models as shown in Table 3. We will incrementally close the gaps and maintain a systematic approach to update the dashboard. We also hope that the dashboard will trigger more models to become available and that metadata for their re-use will be improved.

Finally, the current implementation is limited to single-reference benchmarks and the pipelines assume sentence-level translation. However, multi-reference test sets are extremely rare but we will still consider a support of such data sets in the future. Document-level translation will be important in the near future and for that we will need to adjust our workflow.

Broader Impacts

The OPUS-MT dashboard has the potential to significantly impact the field of MT research by providing a centralized tool for visualizing and evaluating the quality of MT output in a systematic manner. We hope for it to become a point of reference where everyone (1) can consult which model suits best their use case by answering "*Which model should I use for language pair X and domain Y?*" and (2) can obtain proper baselines during paper writing without the need to run again the same experiments, saving time and, more importantly, computational costs. We provide selected rough figures of in-

¹⁷<https://github.com/Helsinki-NLP/OPUS-MT-map>

ference speed in terms of translated benchmarks per GPU hour in Table 3 to illustrate the carbon footprint generated by running the models.

Furthermore, we hope that the overall picture that the OPUS-MT dashboard offers on MT for specific language pairs will encourage the development of resources for low-resource language pairs making it possible to see where there are gaps or where multilingual models fail to deliver.

Acknowledgements

This work was supported by the FoTran project, funded by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under grant agreement No 771113.

This work was also supported by the HPLT project which has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101070350.

References

- Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federmann, Dmitriy Genzel, Francisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, et al. 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*.
- Anastasopoulos Antonios, Barrault Loc, Luisa Bentivogli, Marceley Zanon Boito, Bojar Ondřej, Roldano Cattoni, Currey Anna, Dinu Georgiana, Duh Kevin, Elbayad Maha, et al. 2022. Findings of the iwslt 2022 evaluation campaign. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 98–157. Association for Computational Linguistics.
- Desmond Elliott, Stella Frank, Khalil Sima’an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.
- Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. [Beyond english-centric multilingual machine translation](#).
- Christian Federmann, Tom Kocmi, and Ying Xin. 2022. Ntrex-128–news test references for mt evaluation of 128 languages. In *Proceedings of the First Workshop on Scaling Up Multilingual Evaluation*, pages 21–24.
- Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc’Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. [The Flores-101 Evaluation Benchmark for Low-Resource and Multilingual Machine Translation](#). *Transactions of the Association for Computational Linguistics*, 10:522–538.
- Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc’Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. 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 6098–6111.
- Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. [Marian: Fast neural machine translation in C++](#). In *Proceedings of ACL 2018, System Demonstrations*, Melbourne, Australia.
- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. [Dynabench: Rethinking benchmarking in NLP](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.
- Ondrej Klejch, Eleftherios Avramidis, Aljoscha Burchardt, and Martin Popel. 2015. [Mt-compareval: Graphical evaluation interface for machine translation development](#). *Prague Bull. Math. Linguistics*, 104:63–74.
- Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, and Maja Popović. 2022. [Findings of the 2022 conference on machine translation \(WMT22\)](#). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, Xinyi Wang, and John Wieting. 2019. [compare-mt: A tool for holistic comparison of language generation systems](#). *CoRR*, abs/1903.07926.

- Tommi Nieminen. 2021. [OPUS-CAT: Desktop NMT with CAT integration and local fine-tuning](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 288–294. Online. Association for Computational Linguistics.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022a. [No language left behind: Scaling human-centered machine translation](#).
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022b. [No language left behind: Scaling human-centered machine translation](#).
- Iroko Orife, Julia Kreutzer, Blessing Sibanda, Daniel Whitenack, Kathleen Siminyu, Laura Martinus, Jamiil Toure Ali, Jade Abbott, Vukosi Marivate, Salomon Kabongo, et al. 2020. [Masakhane-machine translation for africa](#). *arXiv preprint arXiv:2003.11529*.
- 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.
- Maja Popović. 2015. [chrF: character n-gram f-score for automatic mt evaluation](#). In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395.
- Maja Popović. 2017. [chrF++: words helping character n-grams](#). In *Proceedings of the second conference on machine translation*, pages 612–618.
- 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.
- 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](#).
- Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. [Comet: A neural framework for mt evaluation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.
- Konstantin Savenkov and Michel Lopez. 2022. [The state of the machine translation 2022](#). 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 32–49, Orlando, USA. Association for Machine Translation in the Americas.
- Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. [Multilingual translation with extensible multilingual pretraining and finetuning](#).
- Jörg Tiedemann. 2020. [The tatoeba translation challenge—realistic data sets for low resource and multilingual mt](#). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182.
- Jörg Tiedemann, Mikko Aulamo, Sam Hardwick, and Tommi Nieminen. 2023. [Open Translation Models, Tools and Services](#), pages 325–330. Springer International Publishing, Cham.
- Jörg Tiedemann and Santhosh Thottingal. 2020. [OPUS-MT – building open translation services for the world](#). In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.
- Jörg Tiedemann. 2012. [Parallel data, tools and interfaces in OPUS](#). In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*, Istanbul, Turkey. European Language Resources Association (ELRA).
- Jörg Tiedemann, Mikko Aulamo, Daria Bakshandaeva, Michele Boggia, Stig-Arne Grønroos, Tommi Nieminen, Alessandro Raganato, Yves Scherrer, Raul Vazquez, and Sami Virpioja. 2022. [Democratizing machine translation with opus-mt](#).
- Leandro von Werra, Lewis Tunstall, Abhishek Thakur, Alexandra Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, Helen Ngo, et al. 2022. [Evaluate & evaluation on the hub: Better best practices for data and model measurement](#). *arXiv preprint arXiv:2210.01970*.

Andy B Yoo, Morris A Jette, and Mark Grondona. 2003. Slurm: Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9*, pages 44–60. Springer.

A Evaluation Hyperparameters

Hyperparameter	Value
beam-size	4
max-length	500
max-length-crop	True
maxi-batch	512
maxi-batch-sort	src
mini-batch	256

Table 4: Marian hyperparameters for decoding internal models.

Hyperparameter	Value
batch_size	64
do_sample	False
max_length	500
num_beams	1
top_k	50

Table 5: Hyperparameters for decoding external models with HuggingFace’s translation pipeline.

B OPUS-MT Map

The OPUS-MT map is yet another visualization of the availability of machine translation models focusing on the language coverage in some geographic distribution. The main purpose is to illustrate the concentration of work on specific regions without making strong claims about the location of specific languages and language speakers. Geolocations are taken from Glottolog and the interactive map supports the visualization of translation for language pairs in both directions using a number of selected benchmarks like the Tatoeba translation challenge and the Flores benchmark. A screenshot of the map is shown in Figure 5.

C Dashboard Screenshots

This appendix shows a number of screenshots from the live dashboard at <https://opus.nlpl.eu/dashboard>. We include different variants of performance plots and show only BLEU-score-based evaluations in the examples shown below. The dashboard makes it possible to show the best performing models for a specific language pair across all benchmarks (see Figure 7), the performance of all models evaluated for a specific benchmark

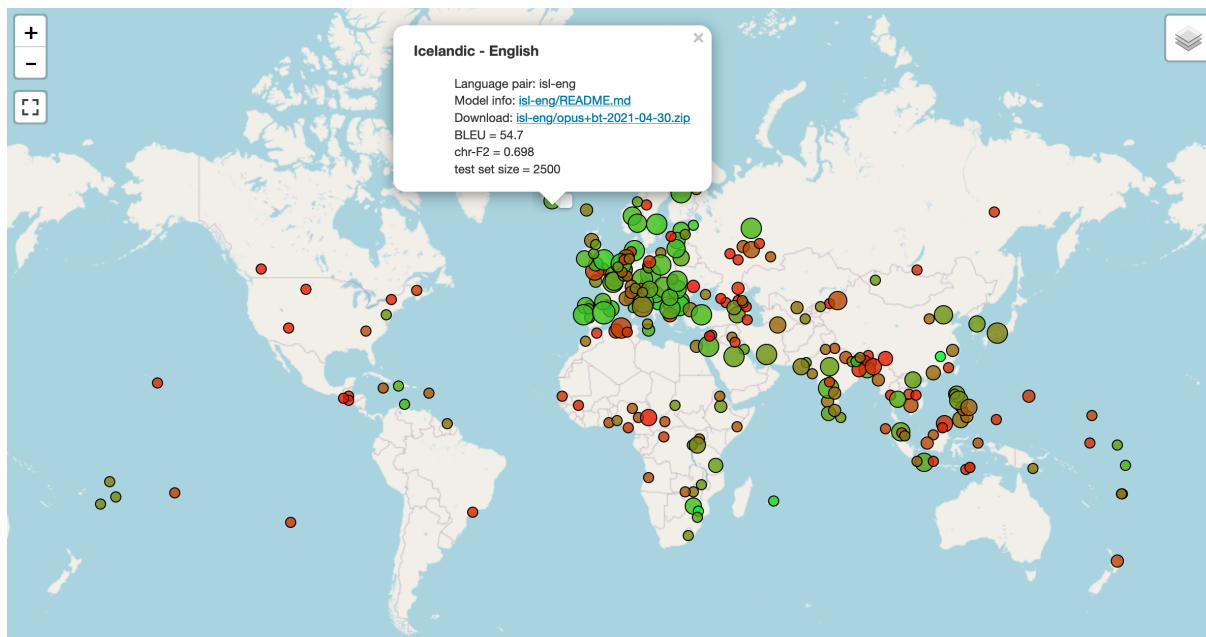


Figure 5: A geographic visualization of the language coverage in public OPUS-MT models. The map plots models according to Tatoeba translation challenge results indicating performance based on color (green is good and red is low performance in BLEU). Smaller circles indicate smaller test sets and are, therefore, less reliable.

(Figure 8), an overview of benchmark results for a selected model including multilingual models (Figure 9) and a comparison of benchmark results for two selected models (Figure 10). An example of the translation inspection feature is shown in Figure 6.

D Video Demonstration

A video demonstration of the system can be accessed at <https://youtu.be/K2cKoAt3AIY>.

- Model: zle-fra/opusTCv20210807_transformer-big_2022-06-24 (model translations in green)
- Test Set: flores200-devtest (reference translations in red)
- Language Pair: ukr-fra
- Show translation without highlighting difference

[start] [show previous] show examples 110 - 119 [show next]

[wdiff][gitdiff][light][dark]

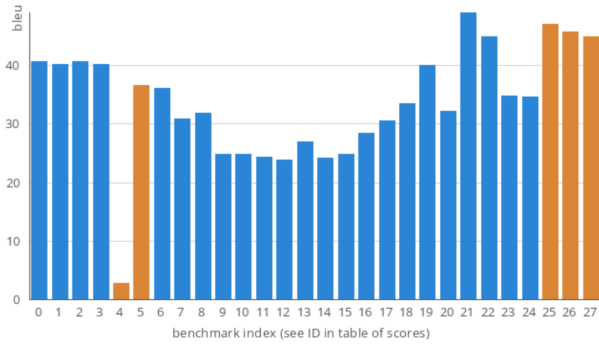
SOURCE: Там вони знайшли тіло Сарожа Баласубраманяна у віці 53 років, вкрите ковдрами у плямах крові.
 REFERENCE: Là, ils ont trouvé le corps de Sarozh Balasubramanian, 53 ans, couvert de couvertures tachées de sang.
 MODEL: Là, ils ont trouvé le corps de Sarozh Balasubramanian à l'âge de 53 ans, couvert de couvertures dans des taches de sang.
 DIFF: Là, ils ont trouvé le corps de [-Sarozha Balasubramanian,-] {+Sarozh Balasubramanian à l'âge de+} 53 ans, couvert de couvertures [-tachées-] {+dans des taches+}

SOURCE: Поліція заявила, що тіло пролежало там близько доби.
 REFERENCE: La police a déclaré qu'il semblerait que le corps se trouvait là depuis un jour environ.
 MODEL: La police a déclaré que le corps était resté là pendant près d'une journée.
 DIFF: La police a déclaré [-qu'il semblerait-] que le corps [-se trouvait-] {+était resté+} là [-depuis un jour environ.-] {+pendant près d'une journée.+}

SOURCE: Про перші випадки хвороби цього сезону було повідомлено наприкінці липня.
 REFERENCE: Les premiers cas de cette maladie saisonnière ont été déclarés fin juillet.
 MODEL: Les premiers cas de cette maladie ont été signalés fin juillet.
 DIFF: Les premiers cas de cette maladie [-saisonnière-] ont été [-déclarés-] {+signalés+} fin juillet.

Figure 6: Browsing through benchmark translations with highlighting differences between reference and system translation. Here a sample from the Flores200 devtest set for Ukrainian to French translated by an OPUS-MT model for East Slavic languages to French.

- Language pair: eng-deu
- Models: [all models] [OPUS-MT] [external] [compare]
- Benchmark: all benchmarks [average score]
- Evaluation metric: bleu [spleu][chrf][chrf+][comet]

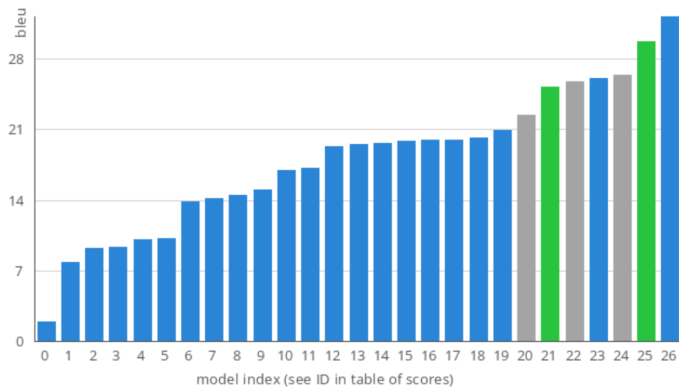


Model Scores (top scoring model on all available benchmarks)

ID	Benchmark	bleu	Output	Model	Link
0	flores101-dev	40.4	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
1	flores101-devtest	40.0	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
2	flores200-dev	40.4	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
3	flores200-devtest	40.0	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
4	multi30k_task2_test_2016	2.7	show	OPUS-MT-models/en-de/opus-2020-02-26	zip-file
5	multi30k_test_2016_flickr	36.3	show	OPUS-MT-models/en-de/opus-2020-02-26	zip-file
6	multi30k_test_2017_flickr	35.8	show	Tatoeba-MT-models/eng-deu/opus-2021-02-22	zip-file
7	multi30k_test_2017_mscoco	30.6	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
8	multi30k_test_2018_flickr	31.6	show	Tatoeba-MT-models/eng-deu/opus-2021-02-19	zip-file
9	news-test2008	24.7	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
10	news2008	24.7	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
11	newssyscomb2009	24.2	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
12	newstest2009	23.6	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
13	newstest2010	26.8	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
14	newstest2011	24.0	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
15	newstest2012	24.7	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
16	newstest2013	28.3	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
17	newstest2014	30.4	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
18	newstest2015	33.3	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
19	newstest2016	39.8	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
20	newstest2017	31.9	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
21	newstest2018	48.8	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
22	newstest2019	44.6	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
23	newstest2020	34.6	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
24	newstestB2020	34.4	show	Tatoeba-MT-models/eng-deu/opus .. 2021-12-08	zip-file
25	tatoeba-test-v2020-07-28	46.7	show	OPUS-MT-models/en-de/opus-2020-02-26	zip-file
26	tatoeba-test-v2021-03-30	45.5	show	OPUS-MT-models/en-de/opus-2020-02-26	zip-file
27	tatoeba-test-v2021-08-07	44.7	show	OPUS-MT-models/en-de/opus-2020-02-26	zip-file

Figure 7: Best performing OPUS-MT models on English-German benchmarks.

- Language pair: eng-ukr
- Models: [all models] [OPUS-MT] [external] [compare]
- Benchmark: [all benchmarks] [average score] flores200-devtest
- Evaluation metric: bleu [spleu][chrf][chrf+][comet]

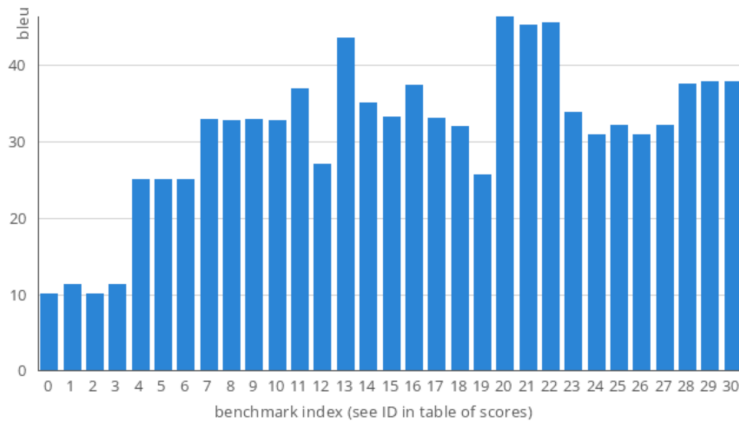


Model Scores (bleu scores on the "flores200-devtest" testset)

ID	bleu	Output	Model	Link
26	32.1	show	Tatoeba-MT-models/eng-zle/opus .. 2022-03-13	zip-file
25	29.6	show	Tatoeba-MT-models/eng-ukr/opus .. 2022-03-23	zip-file
24	26.3	show	facebook/nllb-200-distilled-1.3B	URL
23	25.9	show	Tatoeba-MT-models/eng-ukr/opus .. 2021-04-14	zip-file
22	25.6	show	facebook/nllb-200-1.3B	URL
21	25.1	show	Tatoeba-MT-models/eng-ukr/opus .. 2022-03-05	zip-file
20	22.3	show	facebook/nllb-200-distilled-600M	URL
19	20.8	show	Tatoeba-MT-models/eng-sla/opus .. 2020-08-01	zip-file
18	20.1	show	Tatoeba-MT-models/eng-ukr/opus-2021-02-19	zip-file
17	19.9	show	Tatoeba-MT-models/eng-sla/opus-2020-07-27	zip-file
16	19.8	show	Tatoeba-MT-models/eng-zle/opus .. 2021-04-10	zip-file
15	19.7	show	Tatoeba-MT-models/gmw-zle/opus .. 2021-02-12	zip-file
14	19.5	show	Tatoeba-MT-models/eng-sla/opus-2020-07-06	zip-file
13	19.4	show	Tatoeba-MT-models/eng-sla/opus-2020-07-14	zip-file
12	19.2	show	Tatoeba-MT-models/zle-zle/opus-2020-09-26	zip-file
11	17.1	show	Tatoeba-MT-models/sla-sla/opus-2020-10-04	zip-file
10	16.9	show	Tatoeba-MT-models/sla-sla/opus-2020-09-26	zip-file
9	14.9	show	Tatoeba-MT-models/eng-ine/opus .. 2020-08-01	zip-file
8	14.4	show	Tatoeba-MT-models/eng-ine/opus-2020-07-27	zip-file
7	14.1	show	Tatoeba-MT-models/eng-ine/opus-2020-07-19	zip-file
6	13.8	show	Tatoeba-MT-models/eng-ine/opus-2020-07-14	zip-file
5	10.1	show	Tatoeba-MT-models/eng-mul/opus .. 2020-08-01	zip-file
4	10.0	show	Tatoeba-MT-models/tatoeba-zero .. 2020-06-19	zip-file
3	9.3	show	Tatoeba-MT-models/eng-mul/opus-2020-07-27	zip-file
2	9.2	show	Tatoeba-MT-models/eng-mul/opus-2020-07-20	zip-file
1	7.8	show	Tatoeba-MT-models/eng-mul/opus-2020-07-14	zip-file
0	1.9	show	Tatoeba-MT-models/eng-zle/opus .. 2021-04-16	zip-file

Figure 8: List of models that support translating from English to Ukrainian. Blue bars refer to OPUS-MT models, green bars are compact student models and grey bars refer to external models.

- **Language pair:** [eng-ukr] all languages
- **Models:** [all models] [OPUS-MT] [external] [compare]
- **Selected:** Tatoeba-MT-models/eng-zle/opusTCv20210807+bt_transformer-big_2022-03-13
- **Benchmark:** all benchmarks [download]
- **Evaluation metric:** bleu [spbleu][chrf][chrf+][comet]

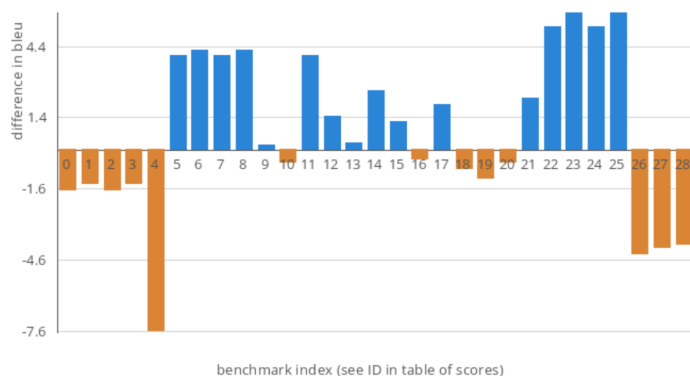


Model Scores (selected model)

ID	Language	Benchmark	Output	bleu	
0	eng-bel	flores101-dev	show	9.9	
1	eng-bel	flores101-devtest	show	11.2	
2	eng-bel	flores200-dev	show	9.9	
3	eng-bel	flores200-devtest	show	11.2	
4	eng-bel	tatoeba-test-v2020-07-28	show	24.9	
5	eng-bel	tatoeba-test-v2021-03-30	show	24.9	
6	eng-bel	tatoeba-test-v2021-08-07	show	24.9	
7	eng-rus	flores101-dev	show	32.8	
8	eng-rus	flores101-devtest	show	32.7	
9	eng-rus	flores200-dev	show	32.8	
10	eng-rus	flores200-devtest	show	32.7	
11	eng-rus	newstest2012	show	36.8	
12	eng-rus	newstest2013	show	26.9	
13	eng-rus	newstest2014	show	43.5	
14	eng-rus	newstest2015	show	34.9	
15	eng-rus	newstest2016	show	33.1	
16	eng-rus	newstest2017	show	37.3	
17	eng-rus	newstest2018	show	32.9	
18	eng-rus	newstest2019	show	31.8	
19	eng-rus	newstest2020	show	25.5	
20	eng-rus	tatoeba-test-v2020-07-28	show	46.3	
21	eng-rus	tatoeba-test-v2021-03-30	show	45.2	
22	eng-rus	tatoeba-test-v2021-08-07	show	45.5	
23	eng-rus	tico19-test	show	33.7	
24	eng-ukr	flores101-dev	show	30.8	
25	eng-ukr	flores101-devtest	show	32.1	
26	eng-ukr	flores200-dev	show	30.8	
27	eng-ukr	flores200-devtest	show	32.1	
28	eng-ukr	tatoeba-test-v2020-07-28	show	37.4	
29	eng-ukr	tatoeba-test-v2021-03-30	show	37.7	
30	eng-ukr	tatoeba-test-v2021-08-07	show	37.7	
				average	30.965

Figure 9: Benchmark results for a multilingual model translating English to East Slavic languages.

- **Model 1 (blue):** Tatoeba-MT-models/eng-zle/opusTCv20210807+bt_transformer-big_2022-03-13
- **Model 2 (orange):** facebook/nllb-200-distilled-1.3B
- **Evaluation metric:** bleu [spbleu][chrf][chrf+][comet]
- **Chart Type:** [standard][diff]



ID	Language	Benchmark (bleu)	Output	Model 1	Model 2	Diff	
0	eng-bel	flores101-dev	compare	9.9	11.6	-1.7	
1	eng-bel	flores101-devtest	compare	11.2	12.6	-1.4	
2	eng-bel	flores200-dev	compare	9.9	11.6	-1.7	
3	eng-bel	flores200-devtest	compare	11.2	12.6	-1.4	
4	eng-bel	tatoeba-test-v2021-08-07	compare	24.9	32.5	-7.6	
5	eng-rus	flores101-dev	compare	32.8	28.8	4.0	
6	eng-rus	flores101-devtest	compare	32.7	28.5	4.2	
7	eng-rus	flores200-dev	compare	32.8	28.8	4.0	
8	eng-rus	flores200-devtest	compare	32.7	28.5	4.2	
9	eng-rus	newstest2012	compare	36.8	36.6	0.2	
10	eng-rus	newstest2013	compare	26.9	27.4	-0.5	
11	eng-rus	newstest2014	compare	43.5	39.5	4.0	
12	eng-rus	newstest2015	compare	34.9	33.5	1.4	
13	eng-rus	newstest2016	compare	33.1	32.8	0.3	
14	eng-rus	newstest2017	compare	37.3	34.8	2.5	
15	eng-rus	newstest2018	compare	32.9	31.7	1.2	
16	eng-rus	newstest2019	compare	31.8	32.2	-0.4	
17	eng-rus	newstest2020	compare	25.5	23.6	1.9	
18	eng-rus	tatoeba-test-v2020-07-28	compare	46.3	47.1	-0.8	
19	eng-rus	tatoeba-test-v2021-03-30	compare	45.2	46.4	-1.2	
20	eng-rus	tatoeba-test-v2021-08-07	compare	45.5	46.0	-0.5	
21	eng-rus	tico19-test	compare	33.7	31.5	2.2	
22	eng-ukr	flores101-dev	compare	30.8	25.6	5.2	
23	eng-ukr	flores101-devtest	compare	32.1	26.3	5.8	
24	eng-ukr	flores200-dev	compare	30.8	25.6	5.2	
25	eng-ukr	flores200-devtest	compare	32.1	26.3	5.8	
26	eng-ukr	tatoeba-test-v2020-07-28	compare	37.4	41.8	-4.4	
27	eng-ukr	tatoeba-test-v2021-03-30	compare	37.7	41.8	-4.1	
28	eng-ukr	tatoeba-test-v2021-08-07	compare	37.7	41.7	-4.0	
				average	31.4	30.6	0.8

Figure 10: Comparison of benchmark results between an OPUS-MT model for English to East Slavic languages and the distilled NLLB model with 1.3B parameters.

The D-WISE Tool Suite: Multi-Modal Machine-Learning-Powered Tools Supporting and Enhancing Digital Discourse Analysis

Florian Schneider^{*†}, Tim Fischer^{*†}, Fynn Petersen-Frey[†]
Isabel Eiser[‡], Gertraud Koch[‡], Chris Biemann[†]

^{*} equal contributions

[†] Language Technology Group, Department of Informatics, Universität Hamburg, Germany

[‡] Institute of Anthropological Studies in Culture and History, Universität Hamburg, Germany
{florian.schneider-1, firstname.lastname}@uni-hamburg.de

Abstract

This work introduces the D-WISE Tool Suite (DWTS), a novel working environment for digital qualitative discourse analysis in the Digital Humanities (DH). The DWTS addresses limitations of current DH tools induced by the ever-increasing amount of heterogeneous, unstructured, and multi-modal data in which the discourses of contemporary societies are encoded. To provide meaningful insights from such data, our system leverages and combines state-of-the-art machine learning technologies from Natural Language Processing and Computer Vision. Further, the DWTS is conceived and developed by an interdisciplinary team of cultural anthropologists and computer scientists to ensure the tool's usability for modern DH research. Central features of the DWTS are: a) import of multi-modal data like text, image, audio, and video b) preprocessing pipelines for automatic annotations c) lexical and semantic search of documents d) manual span, bounding box, time-span, and frame annotations e) documentation of the research process.

1 Introduction

In today's digital era, ever-increasing amounts of heterogeneous, unstructured, and multi-modal data are ubiquitous. Within this data, discourses of contemporary societies are included in various forms, such as news articles or videos, social media postings, forum threads, memes, podcasts, or TV shows. This induces an issue for Digital Humanities (DH) researchers when conducting digital qualitative discourse analysis (Keller, 2011) with such data to examine complex sociological patterns and discussions: It becomes infeasible for a researcher or an average research group to investigate the data manually so they rely on computer assisted qualitative data analysis software (CAQDAS). Although there are many such tools (Eckart de Castilho et al., 2016; Gius et al., 2022; Shnarch et al., 2022; Schneider et al., 2023), they often lack support for such

(amounts of) data or are proprietary software.

With the D-WISE Tool Suite (DWTS) introduced in this work, we provide a novel working environment to support and enhance digital qualitative discourse analysis. The tool is conceived and developed within the D-WISE¹ project in close co-creation by an interdisciplinary team of cultural anthropologists and computer scientists to ensure the tool's usability for modern DH research.

Our system relies on recent advances in Natural Language Processing and Computer Vision and their combination to address the challenges of large amounts of heterogeneous and multi-modal data. Specifically, we employ state-of-the-art text (Devlin et al., 2019; Reimers and Gurevych, 2019), vision (Zhu et al., 2021; Li et al., 2023), speech (Radford et al., 2022), video (Ni et al., 2022; Tong et al., 2022), and visio-linguistic models (Radford et al., 2021) in our multi-modal preprocessing pipeline for tasks like named entity recognition, multi-modal similarity search, object detection, image captioning, automatic speech recognition (transcription), and video understanding tasks. Other essential functionalities for digital discourse analysis like lexical search or manual annotations, are also supported by the DWTS.

In this paper, we describe the system architecture and central features of the DWTS: (a) import of text, image, audio, and video documents; (b) preprocessing pipelines for automatic annotations and indexing; (c) lexical and semantic similarity search, (d) manual annotations; (e) automatic and manual documentation of the research process. The DWTS is designed as an extensible and scalable open-source software system and is publicly available on GitHub². Links to a demonstration instance³ and a video⁴ are provided.

¹ <https://www.dwise.uni-hamburg.de>

² github.com/uhh-lt/dwts

³ acli-dwts.ltdemos.informatik.uni-hamburg.de

⁴ <https://youtu.be/NEmq4AMXVss>

2 Related Work

In the following, we discuss the features and limitations of popular open-source CAQDAS and general purpose annotation tools concerning qualitative discourse analysis on large heterogeneous, multi-modal, and multi-lingual document collections.

WebAnno (Eckart de Castilho et al., 2016), CodeAnno (Schneider et al., 2023), INCePTION (Klie et al., 2018) are a family of web-based collaborative platforms for linguistic annotations based on the UIMA framework. While they support a wide range of annotation-related features for text documents, they do not support image, audio, or video material. Although there exists a WebAnno extension (Remus et al., 2019) for multi-modal document annotations, the tool’s objective is annotating relatively small corpora. Further, the platforms do not include any deep learning technology or preprocessing steps for automatic annotations. Moreover, all platforms lack an overview of the document collection and general search functionality, making it challenging to apply for discourse analysis projects. Another popular tool is CATMA (Gius et al., 2022). Although the web-based platform is designed for qualitative and quantitative research, including annotation, analysis, and visualization functionalities, it only supports text documents. Other CAQDAS systems involving deep learning techniques for automatic annotations are LabelSleuth (Shnarch et al., 2022) and TextAnnotator (Abrami et al., 2020). However, both tools only support textual documents. Recogito2 (Simon et al., 2017) is an open-source tool that can be used for collaborative research projects, is applicable for discourse analysis, and supports text and image annotations. However, the tool does not implement preprocessing pipelines and is designed for small document collections. The widely used CAQDAS tools MAXQDA and Atlas.ti fulfill almost all considered criteria but are proprietary closed-source software, and require expensive licenses, what often poses a hurdle of academic research groups. Further, the tools do not support advanced deep learning features like similarity search, object detection, or automatic audio and video transcriptions.

3 System Demonstration

The D-WISE Tool Suite is a web-based working environment for digital qualitative discourse analysis. It is designed to handle multi-lingual, multi-modal

material of large corpora, achieved by a scaleable architecture built upon a unified data model and state-of-the-art machine learning technology. The application is developed for research groups featuring collaborative projects with multiple users and role management. Further, the system is easily deployable using Docker, not requiring the installation of additional third-party software. Generated data can be exported in common formats for further analysis or processing with other tools.

3.1 Typical Workflow

The D-WISE Tool Suite has a variety of functions that can be used in many diverse ways. For illustrative purposes, we demonstrate the key features with a typical workflow. Imagine Alice, a researcher who examines the discourse on electronic health apps. She accesses DWTS using her web browser and is greeted by a login screen. After registering and logging in, she creates a new project for herself and her project partners.

Data Import To kick off the project, she uploads her material as single files and in a ZIP archive. The data comes in various formats and languages: She saved relevant websites as HTML comprising text and images, related PDF and DOCX documents, and articles in raw text files. Further, she found relevant videos and podcasts on the matter. The DWTS can handle most data formats for text, image, audio, and video documents. Additionally, DWTS offers an external crawler implemented with *Scrapy* and *Beautifulsoup* to scrape websites in case additional material is required.

Data Preprocessing When a document is uploaded, it is automatically pre-processed by the DWTS pipeline. It comprises ML-powered steps to extract metadata and enrich the material with additional information like annotations for named entities or objects in images and videos. It handles vast quantities of multi-modal, multi-lingual data as explained in detail in Section 4.1. While the import operation runs in the background, Alice enjoys a coffee until all data has been processed.

Data Exploration Once the process has finished, Alice starts her research by exploring the data (see Figure 1). The DWTS offers traditional search methods such as full text and keyword search 1). Documents can be further filtered by manually and automatically created annotations, codes and tags 2). During the exploration, Alice can overview the

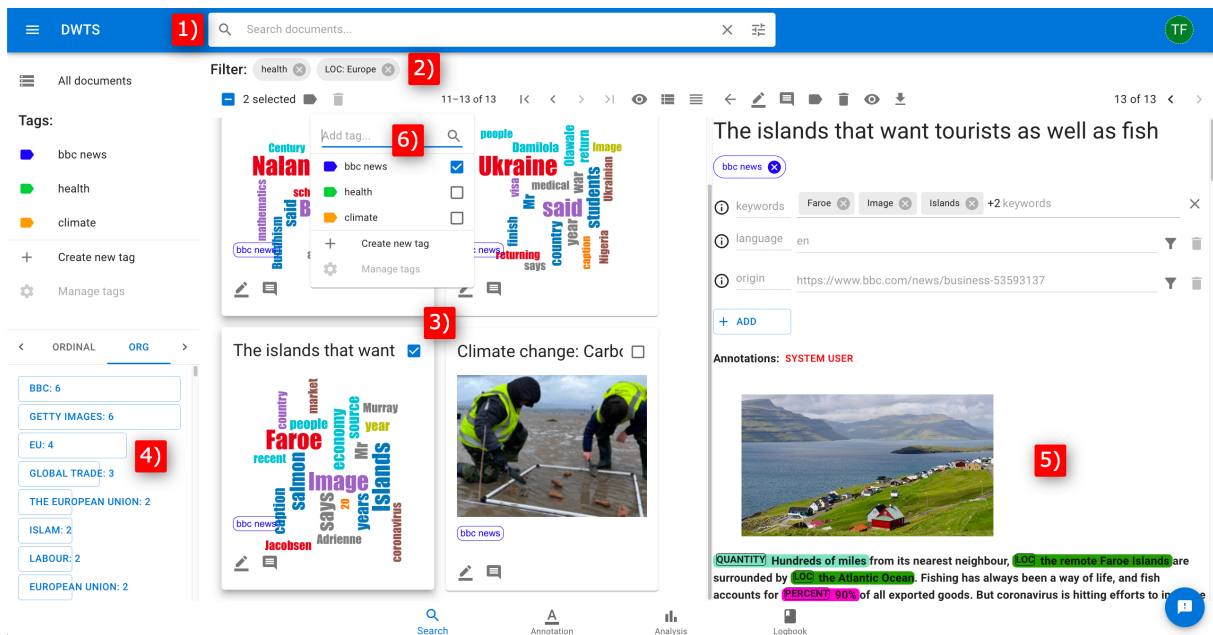


Figure 1: Components of the DWTS search interface: 1) Search bar for lexical search and filter options; 2) Currently applied filters; 3) Multi-modal search results; 4) Search results statistics; 5) Document viewer with tags, metadata, and annotations; 6) Tag editor popup. *The screenshot has been optimized for demonstration purposes.*

current search results 3) by referring to the statistics panel 4) that lists the most frequent keywords, tags, and entities in these documents. Clicking on one of the documents opens a reader view on the right panel 5), which can display text, image, audio, video, or mixed documents. Here, both manual and automatic annotations, e.g., text passages or objects in images or videos, are highlighted. Metadata associated with the document can be viewed and edited as well. This view also allows for multi-modal semantic search by right-clicking on images, videos, or sentences. As documents of all modalities are represented in the same embedding space, the semantic search can be utilized to retrieve similar documents of the same or different modality. Using these functionalities, Alice found several relevant documents related to her research questions. She creates and applies a tag 6) to mark the documents and save them in a collection.

Annotation / Coding Having found relevant documents, Alice decides to read them in detail 1) and opens the annotator (see Figure 2). Using the previously applied tag, she can easily access and jump between those relevant documents with the document explorer 2). While reading, Alice annotates interesting passages in text, audio, and video documents or regions of interest in image documents on the fly 3). During this process, she constructs a

taxonomy by introducing various codes. The code explorer 4) visualizes her team's collaborative hierarchical code tree and allows to rename, delete, and merge codes if required. Codes and annotations from other users and the system can be enabled and disabled 5), which fosters collaboration and discussion. This way, the D-WISE Tool Suite realizes the three coding phases of Grounded Theory (Strauss and Corbin, 1990; Strauss et al., 1996): Open coding by creating new codes on-the-fly and axial and selective coding by providing means to update codes. At any time, Alice can export the automatically and manually created annotations, the created taxonomy, and the raw documents in common formats for further analysis.

Documentation and Reflection While searching and going through documents, Alice learned about new concepts, identified several issues, and developed new ideas and insights on her research topic. The memo feature of the DWTS allows her to attach notes to all objects of interest: documents, annotations, codes, and tags. This reflection process is essential for qualitative discourse analysis to elaborate patterns and phenomena effectively. Finishing her work for today, Alice wants to recap the session. She opens the documentation view, a filterable and searchable overview of her memos. Further, the system automatically logs her interaction



Figure 2: Components of the DWTS annotation interface: 1) The opened document with annotations of the current user and automatic annotations from the SYSTEM; 2) Document explorer with tag selection; 3) Annotation editor popup for code selection; 4) Code explorer with hierarchical codes; 5) Annotation selection; 6) Annotation export button. *The screenshot has been optimized for demonstration purposes.*

with the tool showing exactly when and where she created codes, tags, or annotations. Alice uses the integrated logbook to summarise today’s findings. To share her findings with external researchers, she downloads the logbook and sends it via email.

4 System Architecture

The DWTS is a client-server application with a Python backend and a React frontend. A REST API encapsulates core functionalities to enable communication between frontend and backend.

4.1 Backend Architecture

An essential requirement for the DWTS is scalability, i.e., the system’s ability to operate with large and growing amounts of data and many simultaneous users. This was considered and implemented from the beginning by utilizing only scalable software frameworks and libraries and deploying and orchestrating the system using modern platform-independent technologies. Further, the backend was designed and implemented following common software patterns and idioms to ensure high-quality software fulfilling essential criteria such as extensibility, availability, scalability, and maintainability. Moreover, the system is required to be open-source, i.e., we require third-party software to be open-source licensed, too.

The backend is divided into several components responsible for different functionalities grouped into data storage and retrieval, data preprocessing, communication, and deployment components.

4.1.1 Data Storage and Retrieval

Arguably the most crucial component of the DWTS is its underlying data model. This data model connects the elements of the business logic, e.g., projects, users, memos, or annotations, with the heterogeneous and multi-modal documents in different representations. Business logic data is modeled as SQL tables using the Python ORM framework [SQLAlchemy](#) and stored in a [PostgreSQL](#) database.

Text-, image-, audio-, video-, or mixed-modality documents are represented and stored in several ways. The raw files are stored on disk and can be downloaded by users as described in Section 4.1.3. Semantic vector embeddings of documents or segments are stored in FAISS ([Johnson et al., 2019](#)) indices. To retrieve the best matching documents for a given query, we apply common information retrieval techniques described in more detail in Section 4.1.2. Textual information is stored in inverted indices using [ElasticSearch](#) ([Gormley and Tong, 2015](#))⁵ (ES) and is retrieved via ES Query DSL executed utilizing the [Python ElasticSearch client](#)

⁵ We use v7.16.1 which is open-source licensed

[library](#). A Redis database caches intermediate results from costly operations and stores bug reports and feedback.

4.1.2 Document Preprocessing Pipeline

The document preprocessing pipeline is a central part of the DWTS and responsible for many of its unique selling point features. A schematic overview of the pipeline is shown in Figure 3. In the following, we briefly describe the workflows and essential steps of the pipeline, which is realized as a distributed system using the [Celery framework](#).

Whenever a document is uploaded to the DWTS, a series of actions, referred to as flow, is applied to it depending on its modality. Each flow is executed in an isolated celery worker process that can be run on one or different machines. This design allows for easy scaling of the system if more computing resources are required. Currently, our system supports text, image, audio, and video documents resulting in four different flows, described in more detail in the following.

Text Document Flow Text documents such as HTML, PDF, Word, or TXT files are processed in the text flow of the pipeline. After the document is stored on disk and registered in the database, the textual content gets extracted. For PDF and Word documents, we use [Apache Tika](#). For HTML files, we use [Readability.JS](#), to retain only a website’s content. Image, video, or audio files in an HTML page are extracted beforehand and run separately through the respective flows. Next, metadata like the language of the text is detected using [langdetect](#) to load the language-specific pretrained language model (PLM). Currently, we support English, German, and Italian using the respective transformer models ([Vaswani et al., 2017](#); [Devlin et al., 2019](#)) within the [spaCy framework](#) ([Honni-bal et al., 2020](#)), which we also use in the subsequent steps to do tokenization, sentence segmentation, and Named Entity Recognition. Sentence embeddings are computed using a pretrained multilingual CLIP ([Radford et al., 2021](#); [Reimers and Gurevych, 2020](#)) model from the SentenceTransformers framework ([Reimers and Gurevych, 2019](#)). CLIP is a state-of-the-art visiolinguistic model with strong zero-shot performance in many downstream tasks, including text-to-image and image-to-text retrieval. The embeddings are stored in FAISS ([Johnson et al., 2019](#)) indices. Finally, the textual content is stored in an ElasticSearch index.

Image Document Flow First, the image is stored on disk and in the database with its extracted metadata like the image dimensions. Then, we detect objects in the image using a pretrained DETR ([Zhu et al., 2021](#)) model, an efficient and effective object detection model available within the huggingface transformers framework ([Wolf et al., 2020](#)). Next, a global semantic image embedding is computed using the same CLIP model as in the text flow and stored in a FAISS index. This enables multi-modal similarity search of images and texts. Finally an image caption is generated utilizing a pretrained BLIP-2 ([Li et al., 2023](#)) model. This caption is passed through the text flow to enable lexical and semantic search.

Audio Document Flow In this flow, audio documents in standard formats, such as MP3 or WAV, are processed. The file is stored on disk and in the database along with its extracted metadata like the length of the recording. Then the audio file gets chunked in about 5s segments, which result in about 16.5 words with an average words-per-minute rate of 198 ([Wang, 2021](#)), which is in the range of the average English sentence lengths ([Moore, 2011](#)). These chunks are then forwarded through a Whisper ([Radford et al., 2022](#)) model to compute semantic embeddings stored in a FAISS index to enable audio-to-audio similarity search.⁶ In the final step, we generate a textual transcription of the audio using a Whisper model. We treat the audio transcription as a text document and run it through the text flow to support lexical and semantic textual search.

Video Document Flow Documents in standard video formats, such as MP4 or MOV, are processed in the video flow. Again, first, the file is stored on disk and in the database along with its extracted metadata, like the duration or dimensions. Afterwards, following the motivation for audio files, the video gets chunked into about 5s clips. These chunks are then forwarded through a VideoMAE ([Tong et al., 2022](#)) or X-CLIP ([Ni et al., 2022](#)) model to compute rich semantic embeddings, which are stored in a FAISS index to enable video-to-video similarity search.⁷

Finally, we extract the audio stream from the video and process it in the audio flow.

⁶ The similarity search for raw audio and video documents or chunks is work-in-progress and has yet to be released in our system demonstration DWTS instance. ⁷ This is still undergoing research and is not yet available in the DWTS demonstration instance.

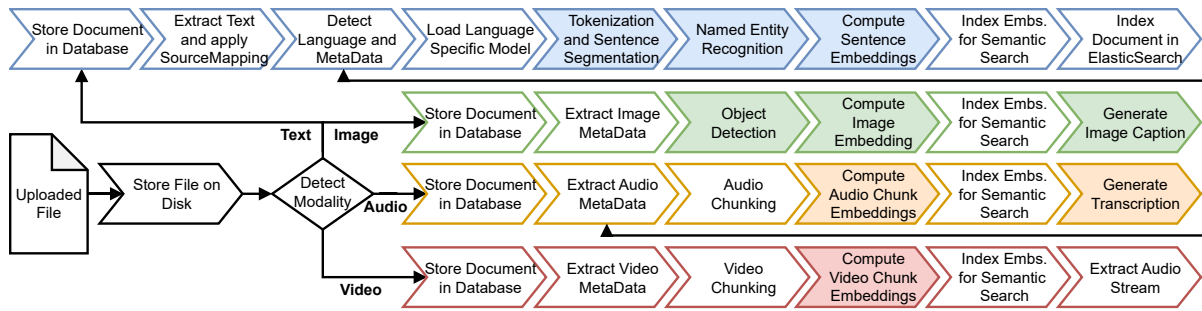


Figure 3: Illustration of the multi-modal preprocessing pipeline in the DWTS. Each flow for a specific modality is executed in a separate Celery worker process. Steps involving deep learning models are highlighted.

4.1.3 Client-Server Communication

A REST API implemented using [FastAPI](#) encapsulates all functionality of the DWTS. Using FastAPI, an [OpenAPIv3](#) schema and a [SwaggerUI](#) are automatically generated, drastically easing the development of UIs or other external applications consuming DWTS functionality. To simplify communication between clients and the server, the parameters and return types of all the API endpoints are defined as data-transfer-objects (DTO) using [Pydantic](#) models. These DTOs are also widely used within the backend to decouple entities from database sessions, for communication between different system components, and to transmit data between the Celery workers of the preprocessing pipeline.

Further, we employ [lighttpd](#) as a file server for raw text, image, audio, and video files.

4.2 Frontend Architecture

The frontend is an interactive web application build with [React](#) and [TypeScript](#). It communicates with the backend by consuming its RESTful API to realize most features. Since the backend exposes its functionality with an API defined by an [OpenAPIv3](#) schema, we use a [code generator](#) to automatically generate a client that fully supports all functionality, including TypeScript interfaces or classes for all parameter and return types. Data fetching, caching, and synchronization are handled with [React-Query](#) making the app faster and more responsive while saving bandwidth and increasing performance. The client state is managed by [Redux](#), which enables powerful functionalities like undo/redo and state persistence. The interface and interaction follow the recognized Material Design System, which utilizes material-like surfaces for the components. We use ready-made components from the [MUI](#) library, as well as custom-tailored components to implement the user interface.

4.3 System Deployment

The DWTS is deployed using modern containerization technology. Every system component, i.e., the databases and indices, API, Celery workers, and the frontend, is deployed in a separate Docker container. All containers are orchestrated in a Docker Compose file that makes up the DWTS system. This approach has several advantages: First, the software is platform-independent and can be conveniently deployed on any modern system or infrastructure with minimal adaptation and configuration efforts. Thus, the system can be deployed on local servers for projects with confidential or privacy constrained data. Second, the components can be deployed on different machines, e.g., computationally intensive components like the Celery workers can run on a GPU server, while memory-intensive components like indices and databases can run on a storage server. Third, the system can be easily scaled using Docker Swarm or similar technology.

5 Conclusion

This paper presented the D-WISE Tool Suite (DWTS), a web-based open-source application to support and enhance digital discourse analysis for Digital Humanities by being able to operate on large amounts of heterogeneous and multi-modal data. We discussed the motivation and need for our system by pointing out the limitations of existing DH tools for extensive multi-modal document collections. Further, we demonstrated central functionalities of the DWTS by describing a typical workflow illustrated by screenshots, and provide technical details about the system architecture in a separate section. Currently work-in-process but released in future work are video and audio annotations and 4-way multi-modal similarity search between text, image, audio, and video documents.

Limitations & Ethics Statement

The DWTS makes state-of-the-art machine learning (ML) models accessible to researchers that could previously not benefit from these advances. Our tool targets Digital Humanities researchers and is intended to assist with qualitative discourse analysis. As with most digital tools, though, it could be misused for other work. We strongly believe that including and enabling more researchers to benefit from modern ML technology outweighs the potential for misuse.

When using ML models, it is important to understand their limitations and critically reflect on their predictions. ML models often include certain biases that can manifest in various types and forms and are not without error. We try to mitigate this by visualizing confidence scores where applicable and additionally provide traditional methods as ML-free alternatives. In particular, we offer a multi-modal semantic similarity search and highlight the confidences, but also provide a standard lexical search to cross-check the results. Still, errors in preprocessing steps, like removing relevant content during data cleanup or the entity recognizer model missing a certain entity of interest, may lead to biased search and analysis results.

Naturally, we also introduce a bias with our system design and envisioned workflow. While we tried our best to model the process of digital discourse analysis as closely as possible, we might still restrict a user in their workflow by our design decisions.

Regarding privacy and security, we identified and mitigated two limitations. The DWTS requires users to upload their potentially sensitive data to the system. To alleviate this, we ensured the tool is easily deployable and can be self-hosted even by non-experts allowing users to stay in full control of their data. In particular, it is even possible to install DWTS locally on private devices. Further, the DWTS includes a feature that automatically logs user actions and populates a logbook in order to improve the documentation and reflection of research processes. By making this an opt-in feature, we guarantee that users are in control of their usage data.

We are aware of the limitations of our system and the technology therein and are committed to actively participating in discussions with domain experts regarding ML, privacy, and bias to identify and iron out further constraints.

Acknowledgement

This work is supported by the D-WISE project, Universität Hamburg, funded by BMBF (grant no. 01UG2124).

References

- Giuseppe Abrami, Manuel Stoeckel, and Alexander Mehler. 2020. TextAnnotator: A UIMA Based Tool for the Simultaneous and Collaborative Annotation of Texts. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 891–900, Marseille, France. European Language Resources Association.
- 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*, pages 4171–4186, Minneapolis, MN, USA.
- Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan.
- Evelyn Gius, Jan Christoph Meister, Malte Meister, Marco Petris, Christian Bruck, Janina Jacke, Mareike Schumacher, Dominik Gerstorfer, Marie Flüh, and Jan Horstmann. 2022. *CATMA: Computer Assisted Text Markup and Analysis*.
- Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide*, 1st edition. O’Reilly Media, Inc.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. <https://spacy.io/>.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.
- Reiner Keller. 2011. The sociology of knowledge approach to discourse (SKAD). *Human Studies*, 34:43–65.
- Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9.

- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. *ArXiv*, abs/2301.12597.
- Andrew Moore. 2011. The long sentence: a disservice to science in the internet age. *BioEssays*, 33(12):193–193.
- Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. 2022. Expanding Language-Image Pretrained Models for General Video Recognition. In *Proceedings of the 17th European Conference on Computer Vision (ECCV)*, pages 1–18, Tel Aviv, Israel.
- 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 (ICML)*, pages 8748–8763, Online.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *ArXiv*, abs/2212.04356.
- 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 3973–3983, Hong Kong, China.
- Nils Reimers and Iryna Gurevych. 2020. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online.
- Steffen Remus, Hanna Hedeland, Anne Ferger, Kristin Bührig, and Chris Biemann. 2019. WebAnno-MM: EXMARaLDA meets WebAnno. In *In Selected papers from the CLARIN Annual Conference, 2018*, Pisa, Italy.
- Florian Schneider, Seid Muhie Yiman, Fynn Petersen-Frey, Gerret von Nordheim, Katharina Kleinen-von Königslöw, and Chris Biemann. 2023. CodeAnno: Extending WebAnno with Hierarchical Document Level Annotation and Automation. In *The 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023), System Demonstrations*, Dubrovnik, Croatia.
- Eyal Shnarch, Alon Halfon, Ariel Gera, Marina Danilevsky, Yannis Katsis, Leshem Choshen, Martin Santillan Cooper, Dina Epelboim, Zheng Zhang, Dakuo Wang, Lucy Yip, Liat Ein-Dor, Lena Dankin, Ilya Shnayderman, Ranit Aharonov, Yunyao Li, Naf-tali Liberman, Philip Levin Slesarev, Gwilym Newton, Shila Ofek-Koifman, Noam Slonim, and Yoav Katz. 2022. Label Sleuth: From Unlabeled Text to a Classifier in a Few Hours. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.
- Rainer Simon, Elton Barker, Leif Isaksen, and Pau de Soto Cañamares. 2017. Linked data annotation without the pointy brackets: Introducing Recogito 2. *Journal of Map & Geography Libraries*, 13(1):111–132.
- Anselm Strauss and Juliet Corbin. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. SAGE Publications, Inc.
- Anselm Strauss, Juliet Corbin, Solveigh Niewiarra, and Heiner Legewie. 1996. *Grounded Theory: Grundlagen Qualitativer Sozialforschung*. Beltz, Psychologie-Verlag-Union Weinheim.
- Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. In *Advances in Neural Information Processing Systems (NIPS)*, New Orleans, LA, USA.
- 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 (NIPS)*, volume 30, pages 5998–6008, Long Beach, CA, USA.
- Li Wang. 2021. British English-Speaking Speed 2020. *Academic Journal of Humanities & Social Sciences*, 4(5):93–100.
- 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 (EMNLP)*, pages 38–45, Online.
- Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *International Conference on Learning Representations (ICLR)*, Vienna, Austria (Virtual Event).

OPENRT: An Open-source Framework for Reasoning Over Tables

Yilun Zhao*¹ Boyu Mi*² Zhenting Qi² Linyong Nan¹

Minghao Guo² Arman Cohan¹ Dragomir Radev¹

¹Yale University ²Zhejiang University

yilun.zhao@yale.edu, miboyu@zju.edu.cn

Abstract

There are a growing number of table pre-training methods proposed for reasoning over tabular data (e.g., question answering, fact checking, and faithful text generation). However, most existing methods are benchmarked solely on a limited number of datasets, varying in configuration, which leads to a lack of unified, standardized, fair, and comprehensive comparison between methods. This paper presents OPENRT, the first open-source framework for reasoning over tabular data, to reproduce existing table pre-training models for performance comparison and develop new models quickly. We implemented and compared six table pre-training models on four question answering, one fact checking, and one faithful text generation datasets. Moreover, to enable the community to easily construct new table reasoning datasets, we developed TARAT, an annotation tool which supports multi-person collaborative annotations for various kinds of table reasoning tasks. The researchers are able to deploy the newly-constructed dataset to OPENRT and compare the performances of different baseline systems. The library OPENRT, along with the annotation tool TARAT, is publicly available at <https://github.com/yilunzhao/OpenRT>.

1 Introduction

With the increasing amount of structured data available, there is a growing interest in developing NLP systems for reasoning over tabular data to perform tasks such as question answering (Pasupat and Liang, 2015; Zhong et al., 2017; Iyyer et al., 2017), fact checking (Chen et al., 2020c; Gupta et al., 2020), and faithful text generation (Chen et al., 2020b; Parikh et al., 2020). Table pre-training has emerged as a promising approach for developing large language models (LLMs) that can perform various kinds of downstream table reasoning

tasks with high accuracy after fine-tuning (Herzig et al., 2020; Liu et al., 2022b; Jiang et al., 2022; Yang et al., 2022; Zhao et al., 2022b; Liu et al., 2022a). However, existing table pre-training methods have been benchmarked on different datasets with varying configurations (Table 2), resulting in a lack of standardization for comprehensive evaluation between methods. Moreover, existing models are developed under individual systems and have a lack of compatibility. Therefore, it is difficult and time-consuming to re-implement them for result comparison in future studies. As the above issues seriously hinder the development of table reasoning models, it is imperative to develop a unified and extensible open-source framework for reasoning over tabular data.

In this paper, we present **OPENRT**, the first **OPEN**-source framework for **R**easoning over **T**abular data, which has the following three characteristics: (1) *Modularization*: we developed OPENRT with highly reusable modules and integrated them in a unified framework, which enables researchers to study different table reasoning models at a conceptual level; (2) *Standardization*: OPENRT includes popular table reasoning datasets and models. The evaluation of different models is standardized under the same experimental configuration; (3) *Extensibility*: OPENRT enables researchers to easily develop their own models or add new datasets by extending corresponding modules with their proposed ones.

Moreover, in order to facilitate the construction of new table reasoning datasets by other researchers, we developed **TARAT**, the first **T**able **R**easoning **A**nnotation **T**ool that supports the collaborative construction of various dataset types (i.e., question answering, fact checking, text generation). User-created datasets can be easily integrated into OPENRT for performance evaluation.

The main structure of the paper is organized as follows: Section 2 describes each table reason-

*Equal Contributions.

Dataset	# Examples	# Tables	Input	Output	Evaluation Metrics
<i>Question Answering</i>					
WIKISQL (Zhong et al., 2017)	80,654	24,241	question	short-form answer	Acc
WTQ (Pasupat and Liang, 2015)	22,033	2,108	question	short-form answer	Acc
SQA (Iyyer et al., 2017)	17,553	982	sequential question	sequential answers	Acc
FeTAQA (Nan et al., 2022a)	10,330	10,330	question	long-form answer	B, R, BS, PARENT, NLI-Acc
<i>Fact Checking</i>					
TABFACT (Chen et al., 2020c)	118,275	16,573	statement	entailment label	Acc
<i>Faithful Table-to-Text Generation</i>					
LOGICNLG (Chen et al., 2020a)	37,015	7,392	highlighted columns	statement	B, R, BS, PARENT, SP/NLI-Acc

Table 1: Table reasoning tasks in OPENRT. B denotes BLEU, R denotes ROUGE, and BS denotes BERTScore. The details of each evaluation metric are introduced in Appendix A.

	WIKISQL	WTQ	SQA	FeTaQA	TABFACT	LOGICNLG
TAPAS (Herzig et al., 2020)	✓	✓	✓		✓	
UnifiedSKG (Xie et al., 2022)	✓	✓	✓	✓	✓	✓
TAPEX (Liu et al., 2022b)	✓	✓	✓	✓	✓	✓
REASTAP (Zhao et al., 2022b)	✓	✓	✓	✓	✓	✓
OmniTab (Jiang et al., 2022)	✓	✓	✓	✓	✓	✓
PLOG (Liu et al., 2022a)	✓	✓	✓	✓	✓	✓

Table 2: The list of table reasoning datasets used in different table pre-training works. It demonstrates the lack of standardized and comprehensive benchmarks for evaluating existing table pre-training methods.

ing task included in OPENRT; Section 3 describes each module and its implementation of OPENRT framework; Section 4 compares the performance of different table pre-training methods on included datasets, and provides insights into how to choose appropriate table pre-training methods for specific needs; Section 5 introduces the functions and implementation of TARAT; finally, Section 6 introduces the related work about table reasoning and annotation tools.

2 OPENRT Tasks

OPENRT covers three kinds of table reasoning tasks: question answering, fact checking, and faithful text generation. The goal of OPENRT is to push the development of table pre-training methods that can be applied and achieved competitive performance on various kinds of table reasoning tasks. We describe the details of each dataset in the following subsections and Table 2.

2.1 Table Question Answering

WIKISQL The WIKISQL-WEAK dataset (Zhong et al., 2017) requires models to perform filtering and, optionally, aggregation on table cell values to obtain an answer to the given question.

WTQ The WikiTableQuestions dataset (Pasupat and Liang, 2015) contains 22,033 complex ques-

tions on Wikipedia tables. Compared to WIKISQL, it requires more complicated reasoning capabilities, thus is more challenging.

SQA The SequentialQA dataset (Iyyer et al., 2017) was built by decomposing the questions from WTQ dataset and organizing them into a conversational context. It requires models to answer sequences of simple but interrelated questions.

FeTAQA Different from above-mentioned three *short-form* Table QA datasets, the Free-form Table Question Answering dataset (Nan et al., 2022b) requires models to generate *free-form* text answers after retrieval, inference, and integration of multiple supporting facts from the source table.

2.2 Table Fact Checking

TABFACT The TABFACT dataset (Chen et al., 2020c) requires the models to perform both soft linguistic reasoning and hard symbolic reasoning to determine whether a given statement is entailed or refuted by the corresponding tabular data.

2.3 Faithful Table-to-Text Generation

LOGICNLG The LOGICNLG dataset (Chen et al., 2020a) requires models to generate multiple statements that perform logical reasoning based on the information in the source table. Each statement

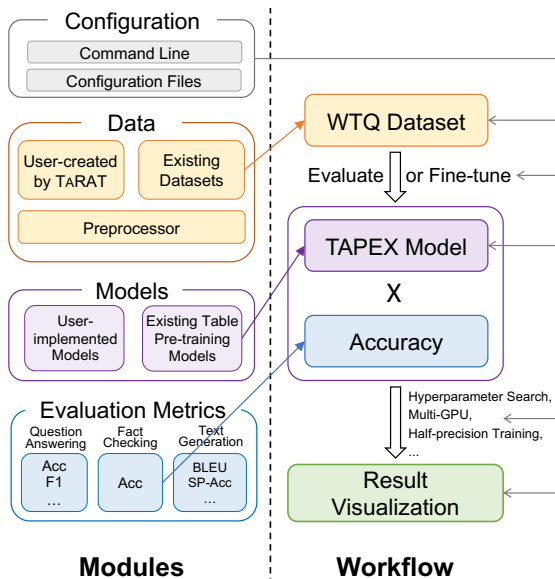


Figure 1: The overall framework of OPENRT.

should be factually correct with the table content.

3 OPENRT Framework

As shown in Figure 1, OPENRT consists of four main modules: configuration, data, modeling, and evaluation. The users are able to fine-tune or test the existing table pre-training models on the included dataset. They are also allowed to add their own models or datasets into OPENRT by extending corresponding modules with their proposed ones.

3.1 Configuration Module

Users and developers define all experiment configurations in the configuration module, which includes command lines, external configuration, and internal configuration. Users are expected to modify the major experiment settings through command lines or by modifying external configuration files, while keeping the internal configuration unchanged for replicating existing models. This ensure a unified and standardized performance comparison between different table reasoning models.

3.2 Data Module

As discussed in Section 2, OPENRT includes popular datasets for table reasoning, which cover various types of tasks. Any raw dataset undergoes processing using the following data flow: raw data \rightarrow Preprocessor \rightarrow Dataset \rightarrow DataLoader \rightarrow processed data. The data flow converts raw datasets in various formats into a unified format that can be used as input for the modeling module.

The *Preprocessor* tokenizes textual and tabular data input using the corresponding tokenizer of the model. It applies the same strategy as Liu et al. (2022b) to truncate a long table into a shorter version to satisfy the model’s input length limit. The *Dataset* component prepares input data, while the *DataLoader* component selects features from the processed data to form tensor data for model input. For both components, we have implemented parent classes `TRDataset` and `TRDataLoader` to include shared attributes and functions. Users can add a new dataset by creating classes that inherit from these parent classes with a few modifications.

3.3 Modeling Module

We have organized and unified the implementations of each table reasoning model within the modeling module by creating an interface parent class called `TRModel`. The design of `TRModel` simplifies the process for users who want to deploy or add a new model to OPENRT. They can simply create and modify a corresponding child class by inherit `TRModel`. The following table reasoning models have been implemented in OPENRT:

- **TAPAS** (Herzig et al., 2020) adopts the BERT encoder with an additional positional embedding for encoding table structure. It also adds two classification layers for cell selection and aggregation operator predictions.
- **UnifiedSKG** (Xie et al., 2022) unifies each task into a text-to-text format, and adopts a sequence-to-sequence T5 model for multi-task learning over multiple table reasoning datasets.
- **TAPEX** (Liu et al., 2022b) pre-trains LLMs by learning as a neural SQL executor to predict the execution results of synthetic SQL queries.
- **REASTAP** (Zhao et al., 2022b) injects various kinds of table reasoning skills (e.g., conjunction, counting) into LLMs by synthesizing Table QA examples as the pre-training corpus.
- **OmniTab** (Jiang et al., 2022) retrieves table-sentence pairs from Wikipedia for mask-based pre-training and synthesizes Table QA examples for pre-training with a QA loss.
- **PLOG** (Liu et al., 2022a) is pre-trained on a synthetic corpus of table-to-logic-form generation to learn table-relevant logical inference knowledge.

While it is possible to train a single model for each task without using the "pre-train, then fine-tune" paradigm (Zhou et al., 2022; Ou and Liu,

	FETAQA				LOGICNLG					
	B-4	ROUGE-1/2/L	BS	NLI	BLEU-1/2/3	ROUGE-1/2/L	BS	PA	SP	NLI
UnifiedSKG	31.5	63.5/41.8/54.1	83.6	78.0	51.8/32.5/18.8	42.8/20.9/36.5	75.1	32.9	46.2	87.0
TAPEX	30.2	62.0/39.9/50.7	82.3	79.2	52.2/32.1/18.3	44.0/21.5/36.8	72.5	31.9	50.1	87.4
REASTAP	30.4	62.5/40.3/51.1	82.7	80.4	52.5/32.5/18.9	44.2/21.5/37.3	78.2	32.2	54.8	89.2
OmniTab	30.7	62.9/40.6/52.1	84.1	81.5	53.0/32.9/19.1	44.5/21.7/37.4	77.6	31.7	55.1	89.0
PLOG	31.8	64.7/42.5/54.9	86.2	80.2	54.9/35.0/21.0	46.1/23.8/39.0	80.1	32.8	50.5	88.9

Table 3: Automated Evaluation of table pre-training models on the test set of FETAQA and LOGICNLG datasets. BS denotes BERTScore, PA denotes PARENT, SP denotes SP-Acc, and NLI denotes NLI-Acc.

	Short-form QA			Fact Checking
	WIKISQL	WTQ	SQA	TABFACT
PLOG	85.9	43.7	60.3	82.0
UnifiedSKG	85.6	48.3	61.5	83.5
TAPAS	84.0	50.4	67.1	81.0
TAPEX	<u>89.2</u>	57.2	74.5	84.0
REASTAP	90.4	<u>58.6</u>	<u>74.7</u>	<u>84.7</u>
OmniTab	88.7	62.8	75.9	85.2

Table 4: Accuracies of existing table pre-training models on the test set of short-form table QA and table fact checking datasets. Bold numbers indicate the highest accuracy, and underscores denote the second best.

2022; Zhao et al., 2023a), we included only *table pre-training models* in OPENRT. This is because we focus on pushing forward the development of more generalizable table pre-training methods that can be applied to various table reasoning tasks and achieve competitive performance.

3.4 Evaluation Module

To evaluate and compare the performance of table reasoning models supported by a certain dataset, OPENRT includes all the evaluation metrics used in the official implementation. These metrics can be used off-the-shelf with a one-line call. The details of each metric are introduced in Appendix A.

3.5 Execution

We implemented *Evaluation* and *Fine-tuning* paradigms for execution in OPENRT. For *Evaluation*, users are able to replicate experimental results of existing models on the supported table reasoning dataset by using provided model checkpoints¹. For *Fine-tuning*, they can train existing models on new datasets or fine-tune their self-implemented models on the included datasets. OPENRT supports

¹We provide checkpoints of each supported model fine-tuned on each included dataset at <https://huggingface.co/OpenTR>

hyper-parameter search to improve fine-tuning performance. We also implemented strategies such as multi-GPU training and half-precision training for efficient model training.

4 Experiments

4.1 Implementation Details

We conducted experiments to evaluate and compare the fine-tuning performance of supported table pre-training models on the included table reasoning datasets. In our experiments, if a model had been fine-tuned on a certain dataset in its original paper and its corresponding checkpoint was publicly available, we evaluated the model’s performance directly using the provided checkpoint. Otherwise, we fine-tuned the model first and then evaluated its performance. For each fine-tuning experiment, we ran 40 epochs with a batch size of 128, and the best fine-tuning checkpoints were selected based on the validation loss.

4.2 Experimental Results

As shown in Table 3, PLOG achieves higher performance for most surface-level evaluations (i.e., BLEU, ROUGE, BERTScore, and PARENT) on faithful table-to-text generation and free-form Table QA tasks. This is reasonable because PLOG is pre-trained to generate logical forms given the tabular data, which improves the model’s capability for content selection and logical inference in text generation. OmniTab achieves the best performance on faithfulness-level evaluation (i.e., SP-Acc and NLI-Acc). It also achieves the best performance on most fact checking and short-form QA tasks (Table 4), demonstrating the effectiveness of pre-training models over natural and synthetic Table QA examples to improve the model’s reasoning capability. Our aim is that such performance comparison, using a standardized benchmark, will provide researchers with valuable insights on how

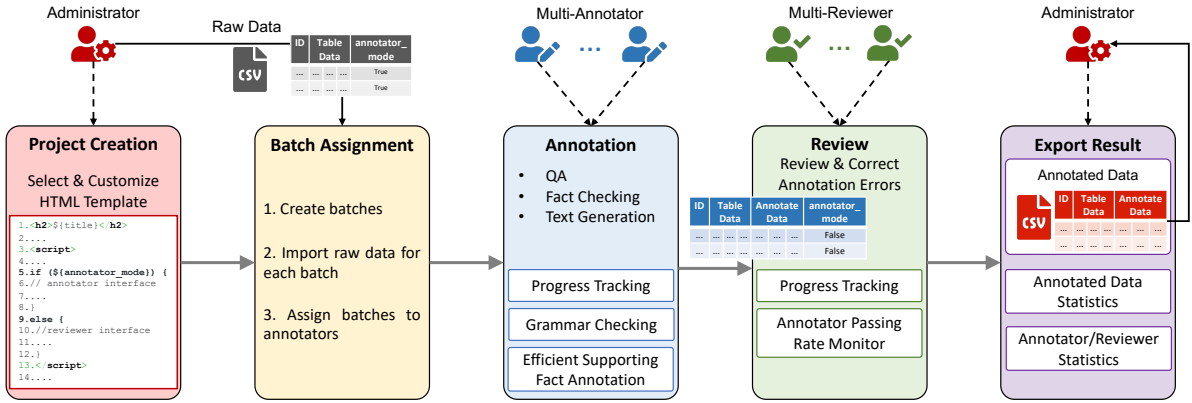


Figure 2: The overall workflow of TARAT.



Figure 3: The four design principles of TARAT: *quick deployment, better quality control, high productivity, and free accessibility*. Each principle comes with a series of feature designs that can make data annotation for table reasoning tasks more efficient and reliable.

to develop more powerful and effective table pre-training methods that can be applied to and achieve competitive performance on various types of table reasoning tasks.

5 TARAT Annotation Tools

In order to facilitate the construction of new table reasoning datasets for other researchers, we developed TARAT, the first open-source table reasoning annotation tool that supports the collaborative construction of various dataset types (i.e., question answering, fact checking, text generation). TARAT was designed, developed, and tested with the four design principles shown in Figure 3. As depicted in Figure 2, a typical annotation process using TARAT consists of the following five steps:

5.1 Annotation Project Creation

The administrator begins by accessing the *admin interface* of TARAT (Figure 4 in Appendix) to specify and set up an annotation project. Specifically, they need to select one of the annotation task templates provided by us as a starting point. These

templates are customizable, so the administrator is allowed to adjust elements (e.g., annotator input type, display style of tabular data) to finalize a tailored annotation task specification.

5.2 Annotation Batch Assignment

The administrator can create multiple batches for an annotation project, with each batch containing multiple annotation tasks (i.e., we count annotating an example as one task). The division of the annotation project into multiple batches helps the administrators better organize and monitor the annotation progress. To initialize each batch, the administrators need to prepare raw annotation data in a csv file, with each line corresponding to an annotation task (Figure 5 in Appendix). Then the administrator can assign each batch to a specific group of annotators (Figure 6 in Appendix).

5.3 Annotation

Once the annotation batches are assigned, the annotators can begin working. In our preliminary study, we found that annotators and reviewers would spend a significant amount of time on typo/grammar correction and table evidence annotation (i.e., write down the row and column indices of relevant table cells). To improve annotation efficiency and quality, we accordingly implemented the following two features:

Grammar Checking We integrated the Grammarly Text Editor Plugin² into the TARAT annotation interface to help annotators detect and eliminate grammar and spelling mistakes. The annotators can view the editing suggestions by clicking the underlined text. They can then apply the sug-

²<https://developer.grammarly.com/docs/>

gested change by clicking “Accept”, or ignore it by clicking “Dismiss” (Figure 9 in Appendix).

Efficient Supporting Fact Annotation Previous work (Chen et al., 2020a, 2021) required annotators to manually write down the column and row indices of all relevant table cells (i.e., supporting fact), which is time-consuming and might introduce typos. To enable a more efficient supporting fact annotation, we implemented *cell highlight*, which allows the annotators to select (i.e., highlight) multiple relevant cells on the table as supporting facts (Figure 10 in Appendix). The indices of highlighted cells will be automatically recorded.

5.4 Annotation Review

Once an annotation batch is finished, the administrator can convert it to a reviewing batch at the TARAT *admin interface*, and assign the reviewing batch to a group of reviewers. The reviewers are expected to correct examples with annotation errors. The system will update the passing rate of each annotator, which the administrator can use to identify unqualified annotators and filter them out.

5.5 Annotation Result Export

After the review process, the annotated data can be exported by the administrator to a result file in CSV format (Figure 8 in Appendix). The administrator is also able to output the annotation statistics (e.g., passing rate, spent time on each example) for each annotator or reviewer, which can be used to determine annotation payment.

6 Related Work

Reasoning over Tabular Data The tasks related to reasoning over tables involves question answering (Pasupat and Liang, 2015; Zhong et al., 2017; Iyyer et al., 2017; Zhao et al., 2022a), fact checking (Chen et al., 2020c; Gupta et al., 2020), and faithful text generation (Chen et al., 2020b; Parikh et al., 2020; Zhao et al., 2023b) based on the information contained in the tables. Previous work mainly investigated how to develop a task-specific model that can work on one or two table reasoning datasets. More recently, inspired by the huge success of pre-trained language models (Devlin et al., 2019; Raffel et al., 2020), researchers have attempted to adopt the "pre-training, then fine-tuning" paradigm to develop models that can handle different kinds of table reasoning tasks with high performance (Herzig et al., 2020; Liu et al.,

2022b; Jiang et al., 2022; Yang et al., 2022; Xie et al., 2022; Liu et al., 2022a). However, existing table pre-training methods have been evaluated on different datasets with varying configurations and developed as individual systems, resulting in difficulties in re-implementing them for performance comparison in future studies. The development of open-source libraries such as *Transformers* (Wolf et al., 2020) alleviate these issues to some extent, but they only cover a narrow range of table pre-training models and datasets. OPENRT implements existing table pre-training models in a unified and highly modularized framework, and provides standardized and comprehensive evaluation benchmarks for performance comparison.

Annotation Tools for Table Reasoning Tasks

Existing annotation tools usually focus on the annotation with only textual input (Nakayama et al., 2018; Perry, 2021; Lin et al., 2022; Friedrich et al., 2022; Pei et al., 2022; Stodden and Kallmeyer, 2022). The development of table-relevant annotation tools is more complex as it requires the system to handle annotations on both textual and tabular input in a user-friendly manner. The current open-source table reasoning annotation tool, TABPERT (Jain et al., 2021), allows a user to update the table contents and associated hypotheses to generate counterfactual NLI examples. Compared to TABPERT, TARAT supports more types of table reasoning tasks, and can be hosted on a centralized server for large-scale distribution with a multi-person collaborative process. Furthermore, each component of TARAT is highly modularized and can be customized to meet the individual needs.

7 Conclusion

This work presents OPENRT, the first open-source framework for reasoning over tabular data, to reproduce existing table pre-training models for a standardized and fair performance comparison. OPENRT also enables users to quickly deploy their own models and datasets. Moreover, we developed TARAT to facilitate the construction of new table reasoning datasets by other researchers.

In the future, we will continue to add more table reasoning datasets and the latest released table pre-training models to OPENRT as part of regular updates. We welcome researchers and engineers to join us in developing, maintaining, and improving OPENRT and TARAT, in order to push forward the development of research on table reasoning.

Acknowledgements

We would like to dedicate this paper to the memory of Dr. Dragomir Radev. Dr. Radev’s leadership, guidance, and expertise were instrumental in shaping the direction and quality of this project. His loss is deeply felt by all of us involved. We extend our heartfelt gratitude to Dr. Radev for his passion and dedication to the whole NLP community.

Ethical Consideration

The datasets included in OPENRT all use licenses that permit us to compile, modify, and publish the original datasets. TARAT is developed based on Turkle³, which is released under the BSD-2-Clause license⁴. Both OPENRT and TARAT are also publicly available with the license BSD-2-Clause, which allows users to modify and redistribute the source code while retaining the original copyright.

References

- Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. [Logical natural language generation from open-domain tables](#). *arXiv preprint arXiv:2004.10404*.
- Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. [Logical natural language generation from open-domain tables](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyong Zhou, and William Yang Wang. 2020c. [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.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Bhuwan Dhingra, 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, Florence, Italy. Association for Computational Linguistics.
- Niklas Friedrich, Kiril Gashteovski, Mingying Yu, Bhushan Kotnis, Carolin Lawrence, Mathias Niepert, and Goran Glavaš. 2022. [AnnIE: An annotation platform for constructing complete open information extraction benchmark](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 44–60, Dublin, Ireland. Association for Computational Linguistics.
- Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. [INFOTABS: Inference on tables as semi-structured data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.
- Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. [Search-based neural structured learning for sequential question answering](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.
- Nupur Jain, Vivek Gupta, Anshul Rai, and Gaurav Kumar. 2021. [TabPert: An effective platform for tabular perturbation](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 350–360, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. [OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.

³<https://github.com/hltcoe/turkle>

⁴<https://opensource.org/licenses/bsd-2-clause/>

- 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.
- Yupian Lin, Tong Ruan, Ming Liang, Tingting Cai, Wen Du, and Yi Wang. 2022. **DoTAT: A domain-oriented text annotation tool**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–8, Dublin, Ireland. Association for Computational Linguistics.
- Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022a. **PLOG: Table-to-logic pre-training for logical table-to-text generation**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. **TAPEX: Table pre-training via learning a neural SQL executor**. In *International Conference on Learning Representations*.
- Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. **doccano: Text annotation tool for human**. Software available from <https://github.com/doccano/doccano>.
- Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022a. **FeTaQA: Free-form table question answering**. *Transactions of the Association for Computational Linguistics*, 10:35–49.
- Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022b. **FeTaQA: Free-form table question answering**. *Transactions of the Association for Computational Linguistics*, 10:35–49.
- Suixin Ou and Yongmei Liu. 2022. **Learning to generate programs for table fact verification via structure-aware semantic parsing**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7624–7638, Dublin, Ireland. Association for Computational Linguistics.
- 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.
- Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. **ToTTo: A controlled table-to-text generation dataset**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.
- Panupong Pasupat and Percy Liang. 2015. **Compositional semantic parsing on semi-structured tables**. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Jiaxin Pei, Aparna Ananthasubramaniam, Xingyao Wang, Naitian Zhou, Apostolos Dedeloudis, Jackson Sargent, and David Jurgens. 2022. **POTATO: The portable text annotation tool**. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 327–337, Abu Dhabi, UAE. Association for Computational Linguistics.
- Tal Perry. 2021. **LightTag: Text annotation platform**. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 20–27, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *Journal of Machine Learning Research*, 21(140):1–67.
- Regina Stodden and Laura Kallmeyer. 2022. **TS-ANNO: An annotation tool to build, annotate and evaluate text simplification corpora**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 145–155, 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.
- Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng

- Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. [UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. [TableFormer: Robust transformer modeling for table-text encoding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland. Association for Computational Linguistics.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#). In *International Conference on Learning Representations*.
- Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022a. [MultiHiert: Numerical reasoning over multi hierarchical tabular and textual data](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.
- Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022b. [ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yilun Zhao, Zhenting Qi, Linyong Nan, Lorenzo Jaime Flores, and Dragomir Radev. 2023a. [Loft: Enhancing faithfulness and diversity for table-to-text generation via logic form control](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.
- Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Xiangru Tang, Yumo Xu, Arman Cohan, and Dragomir Radev. 2023b. [Qt-summ: A new benchmark for query-focused table summarization](#).
- Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2sql: Generating structured queries from natural language using reinforcement learning](#). *arXiv preprint arXiv:1709.00103*.
- Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Fan Cheng, Shi Han, and Dongmei Zhang. 2022. [TaCube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2291, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Appendix

OPENRT includes following evaluation metrics for performance evaluation and comparison:

- **Accuracy** is scored as the number of correct predictions divided by total number of predictions.
- **BLEU** (Papineni et al., 2002) uses a precision-based approach, measuring the n-gram matches between the generated and reference statements.
- **ROUGE** (Lin, 2004) uses a recall-based approach, and measures the percentage of overlapping words and phrases between the generated output and reference one.
- **NLI-Acc** (Chen et al., 2020b) applies a natural language inference (NLI) model fine-tuned on TABFACT (Chen et al., 2020c) to predict whether the generated sentence is entailed by source table.
- **SP-Acc** (Chen et al., 2020b) extracts the meaning representations from the generated sentence and executes them against the source table to verify the logical fidelity of the generated text.
- **BERTScore** (Zhang et al., 2020) computes the similarity between the generated sentence and reference ones using contextual word embeddings from BERT. For LOGICNLG, which has multiple references for a source table, we compute the score by measuring the candidate with each reference and returning the highest score.
- **PARENT** (Dhingra et al., 2019) aligns n-grams from the reference and generated statements to the tabular data before computing their precision and recall. It achieves higher correlation with human judgement.

Figure 4: “Project Creation” in the administrator interface of TARAT. To set up a new annotation project, the administrator needs to choose, modify, and upload the HTML template for initializing the annotation interface.

tableQA.csv ×

appendix > tableQA.csv

Title	Table_link	Question	Answer	Annotation_model
Renaissance (band)	https://raw.githubusercontent.com/yilunzhao/table_csvs/0.csv			true
Mischa Barton	https://raw.githubusercontent.com/yilunzhao/table_csvs/1.csv			true
Triple Crown of Thoroughbred Racing	https://raw.githubusercontent.com/yilunzhao/table_csvs/3.csv			true
1994 European Men's Handball Championship	https://raw.githubusercontent.com/yilunzhao/table_csvs/4.csv			true
Afrikaans	https://raw.githubusercontent.com/yilunzhao/table_csvs/7.csv			true
Geauga County, Ohio	https://raw.githubusercontent.com/yilunzhao/table_csvs/8.csv			true
Oncogene	https://raw.githubusercontent.com/yilunzhao/table_csvs/9.csv			true
Aviation accidents and incidents	https://raw.githubusercontent.com/yilunzhao/table_csvs/10.csv			true
The French Connection (film)	https://raw.githubusercontent.com/yilunzhao/table_csvs/11.csv			true
2nd Wamamaker	https://raw.githubusercontent.com/yilunzhao/table_csvs/12.csv			true

Figure 5: An example of raw data stored in the csv file.

Figure 6: “Annotation Batch Creation” in the administrator interface of TARAT. The administrator can create an annotation batch by importing the raw data stored in a csv file, and assign the batch to a specific group of annotators.

Project: Table QA template / Batch: batch1 Auto-accept next Task Return Task Skip Task Expires in 23:58

Wilco

Year	Award	Work/Artist	Result
1999	Grammy Award for Best Contemporary Folk Album	Mermaid Avenue	Nominated
2005	Grammy Award for Best Alternative Music Album	A Ghost Is Born	Won
2005	Grammy Award for Best Recording Package (awarded to the art director)	A Ghost Is Born	Won
2008	Grammy Award for Best Rock Album	Sky Blue Sky	Nominated
2010	Grammy Award for Best Americana Album	Wilco (The Album)	Nominated
2012	Grammy Award for Best Rock Album	The Whole Love	Nominated

Annotate following:

Question

Answer

Selected areas

Submit

Figure 7: The annotation interface for Table QA task using provided HTML template.

Figure 8: “Annotation Result Export” in the administrator interface of TARAT. The administrator can output the annotated data as well as the annotation statistics in CSV formats.

Annotate following:

Question

Who was the oponent of Derby County in the first game of season?

Add an article ×

... game of **the** season?

The noun phrase **season** seems to be missing a determiner before it. Consider adding an article.

Accept Dismiss ... < >

Grammarly helps you write clearly and mistake-free.

Submit

Figure 9: An example of grammar checking in TARAT. The annotation interface automatically detects the spelling errors and shows the editing suggestions to the annotator.

Hoot Kloot

Nº	Title	Directed by:	Released:
1	"Kloot's Kounty"	Hawley Pratt	1973
2	"Apache on the County Seat"	Hawley Pratt	1973
3	"The Shoe Must Go On"	Gerry Chiniquy	1973
4	"A Self Winding Sidewinder"	Roy Morita	1973
5	"Pay Your Buffalo Bill"	Gerry Chiniquy	1973
6	"Stirrups and Hiccups"	Gerry Chiniquy	1973
7	"Ten Miles to the Gallop"	Arthur Leonardi	1973
8	"Phony Express"	Gerry Chiniquy	1974
9	"Giddy Up Woe"	Sid Marcus	1974
10	"Gold Struck"	Roy Morita	1974
11	"As the Tumbleweeds Turn"	Gerry Chiniquy	1974
12	"The Badge and the Beautiful"	Bob Balsar	1974
13	"Big Beef at O.K. Corral"	Bob Balsar	1974
14	"By Hoot or By Crook"	Bob Balsar	1974
15	"Strange on the Range"	Durward Bonaye	1974
16	"Mesa Trouble"	Sid Marcus	1974
17	"Saddle Soap Opera"	Gerry Chiniquy	1974

Annotate following:

Question

How many movies directed by Gerry Chiniquy were released in the year of 1973?

Answer

3

Selected areas

3:3.2:3;5:6.2:3

Submit

Figure 10: An example of *cell highlight* in TARAT. To annotate supporting facts, the annotators can directly select (i.e. highlight) the relevant table cells on the table. The indices of highlighted cells will be automatically recorded.

UINAUIL: A Unified Benchmark for Italian Natural Language Understanding

Valerio Basile*, Livio Bioglio*, Alessio Bosca**, Cristina Bosco*, and Viviana Patti*

*University of Turin, Italy, email: name.surname@unito.it

**H-FARM, email: alessio.bosca@h-farm.com

Abstract

This paper introduces the Unified Interactive Natural Understanding of the Italian Language (UINAUIL), a benchmark of six tasks for Italian Natural Language Understanding. We present a description of the tasks and software library that collects the data from the European Language Grid, harmonizes the data format, and exposes functionalities to facilitates data manipulation and the evaluation of custom models. We also present the results of tests conducted with available Italian and multilingual language models on UINAUIL, providing an updated picture of the current state of the art in Italian NLU.

Video: <https://www.youtube.com/watch?v=rZWK19cPTbk>

1 Introduction

Large Language Models (LLM) have revolutionized the field of Natural Language Processing. In the span of a few years, the common practice for most NLP tasks shifted from building ad-hoc models trained on task-specific data to fine-tuning general-purpose language models trained in a self-supervised fashion. The focus of the evaluation practices shifted accordingly, from measuring the impact of different features and neural architectures on the prediction performance, to assessing the predictive power of LLMs applied to a variety of NLP tasks. This has been possible, at least in part, due to the standardization proposed in [Devlin et al. \(2019\)](#), where four general task formats are claimed to represent the structure of most NLP tasks: text classification, sentence pair classification, sequence labeling, and question answering.

In this scenario, benchmarks have emerged that collect a number of tasks falling into the four mentioned categories, with the purpose of evaluating LLMs in a fair and reproducible environment. Perhaps the best known of such benchmarks is GLUE ([Wang et al., 2018](#), Language Understanding Evaluation), a set of nine sentence classification and

sentence pair classification Natural Language Understanding tasks. SuperGLUE ([Wang et al., 2019](#)) was presented not long after GLUE with the goal of proposing a harder set of NLU tasks, given the high performance reached by LLMs on GLUE not long after its release.

While English is covered by benchmarks such as GLUE and SuperGLUE, the situation differs sensibly when we turn to other languages, with only a few NLU benchmarks available for non-English languages ([Shavrina et al., 2020](#); [Kakwani et al., 2020](#); [Xu et al., 2020](#); [Wilie et al., 2020](#); [Adesam et al., 2020](#)). These are useful resources for the international NLP community, and a great example of language equality ([Rehm et al., 2021](#)). However, these benchmarks are mostly static collection of datasets with no additional tool to facilitate data gathering and management, evaluation, and automation in general (except for leaderboards, which are usually maintained by the respective authors).

In this paper, we present UINAUIL (Unified Interactive Natural Understanding of the Italian Language), an integrated benchmark for Italian NLU, with three main goals:

- G1 Filling the gap in Italian NLU evaluation in the era of LLMs by proposing one integrated benchmark as opposed to a myriad of individual shared task datasets.
- G2 Raising the bar on the automation level of NLU benchmarks, in order to create more accessible and user-friendly benchmarks.
- G3 Set the example via a use case to encourage scholars and NLP practitioners to publish modern, integrated benchmarks for under-represented languages.

2 A Unified Benchmark for Italian NLP

UINAUIL is a set of six tasks originally proposed as shared tasks for evaluation of Italian NLP. In this

section, we describe the background of the tasks and how they are integrated into UINAUIL.

2.1 EVALITA

In order to select a set of NLU tasks to include into UINAUIL, we analysed the archives of EVALITA. Started in 2007, the “Evaluation campaign for Language Technology in Italian” has been organized every two or three years, with the latest edition currently ongoing in 2023¹. EVALITA provides a common framework of shared tasks, where participating systems are evaluated on a wide variety of NLP tasks. The number of tasks of EVALITA has grown over time, from 5 tasks in the first edition in 2007, to the 14 tasks of the 2020 edition (Pasaro et al., 2020). At the same time, the nature of the proposed tasks has evolved, progressively including a larger variety of exercises oriented to semantics and pragmatics, but without neglecting more classical tasks like part-of-speech tagging and parsing. Since the 2016 edition, EVALITA registered an increased focus on social media data, especially Twitter, and the use of shared data across tasks (Basile et al., 2017).

2.2 EVALITA4ELG

The European Language Grid (ELG) is an European project² whose aim is to establish a platform and marketplace for the European industrial and academic research community around language Technologies (Rehm et al., 2020). ELG is an evolution of META-NET³ and its sister projects T4ME, CESAR, METANET4U, and META-NORD. According to the META-NET Strategic Research Agenda for Multilingual Europe 2020 (Rehm and Uszkoreit, 2013), ELG will represent the “European Service Platform for Language Technologies”. At the time of this writing, ELG counts 7,200 corpora, 3,707 tools and services, plus a large number of other language resources such as lexicons, models, and grammars,⁴ for both EU official and EU candidate languages, as well as a number of non-EU languages, such as languages spoken by EU immigrants or languages of political and trade partners. The platform has an interactive web user interface and APIs. Crucially for the benchmark presented in this paper, a Python library is main-

tained by ELG, which greatly facilitates the programmatic access to its resources.

In 2020, the project EVALITA4ELG “Italian EVALITA Benchmark Linguistic Resources, NLP Services and Tools for the ELG platform”⁵ started, with the aim of collecting all the language resources developed during the several past editions of EVALITA, and integrate them into the ELG. The project succeeded to collect, harmonize and upload 43 corpora and 1 lexical/conceptual resource from all editions of EVALITA from 2007 to 2020 (Basile et al., 2022), among which are found the ones we selected for UINAUIL, described in the next section.

2.3 Tasks

For the first version of UINAUIL, we selected six tasks from EVALITA. We aimed at selecting a representative sample of tasks in terms of their level of language analysis and target phenomenon. Moreover, we selected tasks with different formats, and proposed at different editions of EVALITA. Table 1 summarized the six tasks of UINAUIL, described in detail in the rest of this section.

2.3.1 Textual Entailment

In the textual entailment task of EVALITA 2009 (Bos et al., 2009), participants are asked to submit systems that classify ordered pairs of sentences according to the logical relation holding between them. In particular, a text (T), with respect to an hypothesis (H), can be labeled as either ENTAILED or NOT ENTAILED.

2.3.2 EVENTI

EVENTI, from EVALITA 2014 (Tommaso et al., 2014) is a shared task on Temporal Processing for Italian. The task is built around Ita-TimeBank (Caselli et al., 2011), a large manually annotated dataset of events and temporal expressions. The dataset follows the TimeML tagging standard, where events and time expressions are labeled as spans of single or multiple tokens, and they may be associated with attributes. The shared task is articulated into four subtasks related to the prediction of the extent of events and timex, their classification, and the relations insisting between event/event or event/timex pairs.

In UINAUIL, we include the subtask B, which involves the tagging of events and their classification

¹<http://www.evalita.it/>

²<https://cordis.europa.eu/project/id/825627>

³<http://www.meta-net.eu/>

⁴<https://live.european-language-grid.eu/>

⁵<http://evalita4elg.di.unito.it/>

Acronym	Full name	Task type	Size (training/test)
Textual Entailment	Textual Entailment	Sentence pair classification	400/400
EVENTI	Event detection & classification	Sequence labeling	5,889/917
FactA	Factuality classification	Sequence labeling	2,723/1,816
SENTIPOLC	Sentiment Polarity Classification	Sentence classification	7,410/2,000
IronITA	Irony Detection	Sentence classification	3,777/872
HaSpeeDe	Hate Speech Detection	Sentence classification	6,839/1,263

Table 1: Summary of the tasks included in UINAUIL.

according to one of the following classes: ASPECTUAL (phase or aspect in the description of another event); I_ACTION (intensional action); I_STATE (event that denotes stative situations which introduce another event); PERCEPTION (event involving the physical perception of another event); REPORTING (action of declaring something, narrating an event, informing about an event); STATE (circumstance in which something obtains or holds true); OCCURRENCE (other type of event describing situations that happen or occur in the world).

The task is a sequence labeling problem, therefore the classes are associated at the token level, prefixed with a B (beginning of a labelled span) or a I (inside a labelled span), while O (outside) denotes the tokens that do not belong to any event.

2.3.3 FactA

The Event Factuality Annotation (FactA) task was part of EVALITA 2016 (Minard et al., 2016). In this task, the participant systems are challenged to profile the factuality of events in the text by means of three attributes, namely: certainty, time, and polarity. In UINAUIL, we included the first subtask of FactA, that is, the labeling of certainty, whereas spans of tokens from the input text associated with an event are labeled with one of three classes: CERTAIN (the source is certain about the mentioned event); NON_CERTAIN (the source is not certain about the mentioned event); UNDERSPECIFIED (the certainty about the mentioned event is not specified). Like EVENTI, FactA is a sequence labeling task and therefore the annotation is at the token level following the BIO standard.

2.3.4 SENTIPOLC

The SENTIment POLArity Classification (SENTIPOLC) shared task was proposed for the first time at EVALITA 2014 (Basile et al., 2014) and then re-run in 2016 (Basile et al., 2014). SENTIPOLC is divided into three subtasks, two of

which are binary classification tasks where systems are challenged to predict subjectivity and irony in Italian tweets. The other task is polarity prediction, where systems have to predict two independent binary labels for positivity and negativity, with all four possible combinations of values allowed. The polarity prediction task is included in UINAUIL, with a slight change of format: instead of two independent binary labels, the task in UINAUIL is cast as a four-value multiclass classification task with labels POSITIVE, NEGATIVE, NEUTRAL, and MIXED.

2.3.5 IronITA

The shared task on Irony Detection in Italian Tweets (Cignarella et al., 2018, IronITA) is a shared task focused on the automatic detection of irony in Italian tweets, from EVALITA 2018. The task comprises two subtasks with differing by their level of granularity. The first subtask is a binary classification of tweets into ironic vs. non-ironic. The second task adds the level of sarcasm to the classification, conditioned on the presence of irony in the tweets. In UINAUIL, we included the first task, as a sentence-level binary classification task with the labels IRONIC and NOT IRONIC.

2.3.6 HaSpeeDe

Hate Speech Detection (HaSpeeDe) from EVALITA is a classification task from EVALITA 2020 (Sanguinetti et al., 2020), updated re-run of the same task from EVALITA 2018 (Bosco et al., 2018). The task invites participants to classify social media data from Twitter and Facebook as hateful, aggressive, and offensive. The complete shared task comprises binary classification (HATE vs. NOT HATE), a cross-domain subtask, stereotype detection and the identification of nominal utterances linked to hateful content. In UINAUIL, we included the training and test data for the main classification task only.

3 The UINAUIL library

In addition to defining the benchmark, we created a software library to access the data and functionalities of UINAUIL. We developed a Python module for downloading the task data and metadata, represented in a structurally consistent way. Furthermore, the library provides an implementation of evaluation metrics. The UINAUIL package is available on pip / PyPI, and can be installed with the command:

```
1 $ pip install uinauil
```

Listing 1: How to install the UINAUIL package.

The package depends only on two non-standard packages: `elg`, the library maintained by the ELG project for accessing to the resources of the European platform ⁶ (see Section 2.2), and `scikit-learn`, a well known library for Machine Learning and Data Analysis ⁷. Otherwise, since the UINAUIL library is fully contained into a single file, developers can directly download the source file and save it in the working folder.

Once installed, the library can be used to download the resources of the tasks described in Section 2.3, and to evaluate the predictions of a personal model through standard performance metrics selected by us for each task. The data are contained in Python standard structures (lists and dictionaries) and are divided into training and test sets, according to the original split for each task represented in the ELG repository. The list of available tasks is stored into a proper attribute:

```
1 >>> import uinauil as ul
2 >>> ul.tasks
3
4 {
5   'haspeede': {
6     'id': 7498,
7     'task': 'classification'
8   },
9   'textualentailment': {
10    'id': 8121,
11    'task': 'pairs'
12  },
13  'eventi': {
14    'id': 7376,
15    'task': 'sequence'
16  },
17  'sentipolc': {
18    'id': 7479,
19    'task': 'classification'
20  },
21  'facta': {
22    'id': 8045,
```

⁶<https://pypi.org/project/elg/>

⁷[scikit-learn.org/](https://pypi.org/project/scikit-learn/)

```
23   'task': 'sequence'
24 },
25 'ironita': {
26   'id': 7372,
27   'task': 'classification'
28 }
```

Listing 2: List of available tasks.

The `tasks` variable contains information in a dictionary format, where each key is the name of a task (used to access the task data in UINAUIL), while the value contains its identifier on the ELG platform and the type of task. An example of usage of the library for a learning and evaluation pipeline is the following:

```
1 import uinauil as ul
2
3 # load a task, for example 'facta'
4 task = ul.Task('facta')
5
6 # get training and test set of the task
7 train = task.data.training_set # train
8 test = task.data.test_set     # test
9
10 # train the model on the training set
11 # and make prediction on test set
12 ...
13 pred = <make predictions on test set>
14
15 # evaluate model on standard metrics
16 scores = task.evaluate(pred)
17 print(scores)
```

Listing 3: Quickstart for UINAUIL package.

Line 1 imports the UINAUIL library, while Line 4 downloads the resources of a task, in the example the FactA task described in Section 2.3.3. The authentication on ELG is handled by the `elg` library and is equipped with a caching mechanism in order to minimize the requests for logins on the platform. The UINAUIL library also checks whether the data was previously downloaded before connecting to ELG. Lines 7 and 8 store the training and test sets in local variables. These are represented as lists of instances for classification tasks, or lists of lists of tokens for sequence labeling tasks. In turn, instances and tokens are dictionaries pairing text and labels. At this point of the code example, a model can be trained on the training data, or the labels can be predicted otherwise, depending on the implemented approach — the library is agnostic to specific classification models. On Line 15 the predictions are used to evaluate the model with the `evaluate` method of UINAUIL, that calculates several standard performance metrics chosen specifically for each task as follows:

- For sequence labeling tasks, the performance

is evaluated only with accuracy, calculated as the ratio of hits over all the tokens.

- For all the remaining tasks, the performance metrics are accuracy on all classes, then precision, recall and F1 for each single class and their macro average.

In addition to these core functionalities, the UIN-AUIL library contains several metadata that helps programmers to understand the resources of each task, including the list of key names of features and target, a brief description of the meaning of each feature, the list of possible values of the target and their meaning, and others. The complete list of variables and methods of UINAUIL library is available on the Github repository of the project⁸. Also present on the repository are several examples of use of the library on several common Machine Learning models in form of Python notebooks, and the complete leaderboards by task.

4 Evaluation

In order to test the library, and to offer the community a first set of results on Italian NLU tasks, we conducted a series of experiments with the aim of setting a baseline for all the tasks of the benchmark. The experiments consist in fine-tuning pre-trained language models for Italian (plus a multilingual one) on the training data of each task, and testing their prediction against the corresponding test data, computing the appropriate evaluation metrics.

4.1 Experimental setting

We implemented this series of experiments with *simpletransformers*⁹, a Python library that facilitates LLM fine-tuning and prediction. Simpletransformers automatically downloads pre-trained models from the Huggingface repository¹⁰, and provides functions for training and classification. We built scripts that collect data through the UIN-AUIL library, fine-tune LLMs, produce the predictions with simpletransformers, and finally use UINAUIL again to compute the relevant evaluation metrics. We kept the hyperparameter optimization at a minimum, on purpose, since the goal of these experiments is not that of achieving a high performance, as much as producing a fair (while still high) baseline, and a comparison between models

⁸<https://github.com/valeribasile/uinauil>

⁹<https://simpletransformers.ai/>

¹⁰<https://huggingface.co/models>

across tasks. All models are fine-tuned for exactly 2 epochs, with a fixed learning rate of 10^{-4} . All experimental results are averages of five runs.

4.2 Models

Here we briefly describe the LLMs used in the baseline experiments. The string in brackets is the identifier of the model in Huggingface.

- ALBERTO
(m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0) is the first LLM that has been proposed for the Italian language (Polignano et al., 2019). This model is based on BERT and it is trained on a collection of 200 million posts from Twitter from TWITA (Basile et al., 2018).
- ITALIAN BERT
(dbmdz/bert-base-italian-uncased, dbmdz/bert-base-italian-xxl-uncased) is a LLM maintained by the MDZ Digital Library at Bavarian State, based on ELECTRA (Clark et al., 2020) and trained on a Wikipedia dump, the OPUS corpora collection (Tiedemann and Nygaard, 2004), and the Italian part of the OSCAR corpus (Abadji et al., 2021) for a total of about 13 million tokens. The model comes in two variants, the regular one and a larger one (XXL).
- MULTILINGUAL BERT
(bert-base-multilingual-uncased) is one of the first models released together with the BERT architecture itself (Devlin et al., 2019). It is trained on text in 102 languages from Wikipedia with a masked language model goal. Although it has been surpassed in performance for many NLP tasks, Multilingual BERT has been widely adopted, also because pre-trained language models for languages other than English are often unavailable or smaller than their English counterparts.

4.3 Results

Table 2 shows the results of the baseline systems on the UINAUIL tasks. Focusing on the sequence labeling tasks EVENTI and FactA, we notice how the model does not make substantial difference for the latter (factuality classification), while there is a 0.02 point difference in performance for the former (event classification). The larger model (Italian BERT XXL) is the one obtaining the best

Model	Textual Entailment				SENTIPOLC				EVENTI
	P	R	F1	Acc.	P	R	F1	Acc.	Acc.
ITALIAN BERT	.441	.497	.404	.538	.741	.721	.716	.646	.916
ITALIAN BERT XXL	.391	.495	.379	.541	.764	.741	.740	.675	.936
ALBERTO	.427	.500	.391	.529	.727	.688	.691	.621	.925
MULTILINGUAL BERT	.445	.524	.430	.544	.660	.653	.645	.559	.925

Model	IronITA				HaSpeeDe				FactA
	P	R	F1	Acc.	P	R	F1	Acc.	Acc.
ITALIAN BERT	.737	.736	.735	.736	.786	.785	.785	.785	.907
ITALIAN BERT XXL	.769	.765	.764	.765	.792	.791	.791	.791	.908
ALBERTO	.744	.743	.742	.742	.744	.742	.741	.741	.909
MULTILINGUAL BERT	.710	.709	.709	.709	.743	.740	.739	.739	.909

Table 2: Baseline results on all task of UINAUIL project: Textual Entailment, SENTIPOLC, IronITA and HaSpeeDe in terms of macro-averaged precision (P), recall (R), and F1-score (F1), and accuracy; EVENTI and FactA in terms of token-level accuracy.

performance on EVENTI, as well as on all the sentence classification tasks SENTIPOLC, IronITA, and HaSpeeDe.

Interestingly, for Textual Entailment, Multilingual BERT is the best model. This is also the only sentence pair classification task of the benchmark, indicating how the pre-training strategy of LLMs (e.g., stronger emphasis on single text vs. sentence pair) has an impact on its performance on different tasks.

In absolute terms, the performances of all baselines on Textual Entailment are quite poor, with an accuracy slightly higher than 0.5 and a very low F1 score, around 0.4. This shows how even if this task has been published over a decade ago, there is still ample room for improvement.

The performance on the baselines on the classification tasks are all in line with the reported state of the art, validating the standardization proposed with our benchmark.

5 Conclusions

In this paper we presented UINAUIL (Unified Interactive Natural Understanding of the Italian Language), an integrated benchmark for Italian NLU. Its purposes are manifold: to fill the gap in Italian NLU evaluation by proposing one integrated benchmark; to create more accessible and user-friendly benchmarks for Italian NLU; to encourage scholars to publish modern, integrated benchmarks for under-represented languages.

UINAUIL is implemented in Python library, publicly available via pip/PyPI, that permits to easily

download resources in Italian Language for six different NLU tasks, that can be used by programmers and researchers to train and evaluate their NLP models. UINAUIL is built with automation as principle, with the main goal of minimizing the overhead for a user who wants to evaluate a NLU model for Italian. In effect, only a few lines of code are sufficient before and after the main logic of a model, in order to retrieve the data and evaluate the model.

In this paper, we presented in details each task included into UINAUIL and the main features of the Python library, including a sample quickstart code for its most common functionalities. Furthermore, we evaluated the performances of several common NLP models for Italian Language on each task of UINAUIL. The results show that current models represent a high-performing baseline, especially for sentence classification tasks, while there is still room for improvement for Italian NLU, as shown by the performance on the textual entailment task. However, it should be noted that the goal of this paper is mainly to present the UINAUIL benchmark. A thorough analysis of the results of all available models, while out of our present scope, is a natural next development of this work.

As further developments, we plan to add other tasks to the project, accordingly to the future developments in the field of NLP for Italian Language. We also plan to implement the leaderboard as a service in ELG, besides the Github repository, so that users can submit their results autonomously, leveraging the provided authentication.

6 Ethical and legal statement

This work uses data that have been previously reviewed and published. As such, we find no particular ethical issue to be discussed beyond what is already discussed by the original articles presenting the datasets. One particular dataset, however, contains sensitive data: the HaSpeeDe shared task data made of tweets annotated for hate speech. While the user mentions in these tweets were anonymized by the authors of the dataset to protect the mentioned people’s privacy, the texts still contain explicit and implicit expressions of hatred that may result hurtful to some readers.

The download of the datasets is managed through the European Language Grid. As part of the procedure, the user is informed about the terms and conditions of each individual dataset and must accept the licence before downloading the data. Furthermore, the ELG platform tracks the data exchange in order to comply with the European General Data Protection Regulation (GDPR).

7 Limitations

In this paper, in addition to a benchmark for Italian NLU and a Python library implementing it, we presented the results of pre-trained language models fine-tuned for the six tasks in the benchmark. The models we selected are widely used for the Italian language, but they are not the only available ones. Moreover, the scenario moves fast, with newer and larger language models being published regularly. The evaluation conducted in this paper, therefore, can only be a partial snapshot of the current panorama, while the UINAUIL library stands as an easy tool to evaluate new models as they are published with minimal overhead.

8 Acknowledgments

This work was partially supported by the EVALITA4ELG project (Italian EVALITA Benchmark Linguistic Resources, NLP Services and Tools for the ELG platform), funded by the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 825627 (ELG). We further would like to thank Bernardo Magnini for the insightful discussions leading to the current shape of UINAUIL.

References

- Julien Abadji, Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2021. [Ungoliant: An optimized pipeline for the generation of a very large-scale multilingual web corpus](#). In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-9) 2021. Limerick, 12 July 2021 (Online-Event)*, pages 1 – 9, Mannheim. Leibniz-Institut für Deutsche Sprache.
- Yvonne Adesam, Aleksandrs Berdicevskis, and Felix Morger. 2020. Swedishglue – towards a swedish test set for evaluating natural language understanding models. Technical report, University of Gothenburg.
- Pierpaolo Basile, Viviana Patti, Francesco Cutugno, Malvina Nissim, and Rachele Sprugnoli. 2017. Evalita goes social: Tasks, data, and community at the 2016 edition. *Italian Journal of Computational Linguistics*, 3-1.
- Valerio Basile, Andrea Bolioli, Viviana Patti, Paolo Rosso, and Malvina Nissim. 2014. Overview of the evalita 2014 sentiment polarity classification task. *Overview of the Evalita 2014 SENTiment POLarity Classification Task*, pages 50–57.
- Valerio Basile, Cristina Bosco, Michael Fell, Viviana Patti, and Rossella Varvara. 2022. [Italian NLP for everyone: Resources and models from EVALITA to the European language grid](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 174–180, Marseille, France. European Language Resources Association.
- Valerio Basile, Mirko Lai, and Manuela Sanguinetti. 2018. [Long-term social media data collection at the university of turin](#). In *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Torino, Italy, December 10-12, 2018*, volume 2253 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS.org.
- Johan Bos, Fabio Massimo Zanzotto, and Marco Panacchiotti. 2009. Textual entailment at evalita 2009. In *Poster and Workshop Proceedings of the 11th Conference of the Italian Association for Artificial Intelligence*, volume 9, Reggio Emilia, Italy.
- Cristina Bosco, Felice Dell’Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 hate speech detection task. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*, volume 2263, pages 1–9, Torino. CEUR Workshop Proceedings (CEUR-WS.org).
- Tommaso Caselli, Valentina Bartalesi Lenzi, Rachele Sprugnoli, Emanuele Pianta, and Irina Prodanof. 2011. [Annotating events, temporal expressions and relations in Italian: the it-timeml experience for the](#)

- ita-TimeBank. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 143–151, Portland, Oregon, USA. Association for Computational Linguistics.
- Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the Evalita 2018 task on Irony Detection in Italian Tweets (IRONITA). In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*, volume 2263, pages 1–9, Torino. CEUR Workshop Proceedings (CEUR-WS.org).
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. **Electra: Pre-training text encoders as discriminators rather than generators**. In *International Conference on Learning Representations*, pages 1 – 18.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. **IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.
- Anne-Lyse Minard, Manuela Speranza, Tommaso Caselli, and Fondazione Bruno Kessler. 2016. The EVALITA 2016 event factuality annotation task (FactA). In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016)*, volume 1749, Napoli. CEUR Workshop Proceedings (CEUR-WS.org).
- Lucia C Passaro, Maria Di Maro, Valerio Basile, and Danilo Croce. 2020. Lessons learned from evalita 2020 and thirteen years of evaluation of italian language technology. *IJCoL. Italian Journal of Computational Linguistics*, 6(6-2):79–102.
- Marco Polignano, Pierpaolo Basile, Marco de Gemmis, Giovanni Semeraro, and Valerio Basile. 2019. **Alberto: Italian BERT language understanding model for NLP challenging tasks based on tweets**. In *Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019*, volume 2481 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS.org.
- Georg Rehm, Maria Berger, Ela Elsholz, Stefanie Hegele, Florian Kintzel, Katrin Marheinecke, Stelios Piperidis, Miltos Deligiannis, Dimitris Galanis, Katerina Gkirtzou, Penny Labropoulou, Kalina Bontcheva, David Jones, Ian Roberts, Jan Hajič, Jana Hamrlová, Lukáš Kačena, Khalid Choukri, Victoria Arranz, Andrejs Vasiljevs, Orians Anvari, Andis Lagzdīns, Jūlija Melņika, Gerhard Backfried, Erinc Dikici, Miroslav Janosik, Katja Prinz, Christoph Prinz, Severin Stampler, Dorothea Thomas-Aniola, José Manuel Gómez-Pérez, Andres Garcia Silva, Christian Berrío, Ulrich Germann, Steve Renals, and Ondrej Klejch. 2020. **European language grid: An overview**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3366–3380, Marseille, France. European Language Resources Association.
- Georg Rehm, Federico Gaspari, German Rigau, Maria Giagkou, Stelios Piperidis, Annika Grützner-Zahn, Natalia Resende, Jan Hajic, and Andy Way. 2021. The european language equality project: Enabling digital language equality for all european languages by 2030. *The Role of National Language Institutions in the Digital Age*, page 17.
- Georg Rehm and Hans Uszkoreit. 2013. *META-NET strategic research agenda for multilingual Europe 2020*. Springer.
- Manuela Sanguinetti, Gloria Comandini, Elisa Di Nuovo, Simona Frenda, Marco Stranisci, Cristina Bosco, Tommaso Caselli, Viviana Patti, and Irene Russo. 2020. **HaSpeeDe 2@EVALITA2020: Overview of the EVALITA 2020 Hate Speech Detection Task**. In *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020)*, pages 1–9, Online. CEUR Workshop Proceedings (CEUR-WS.org).
- Tatiana Shavrina, Alena Fenogenova, Emelyanov Anton, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. **RussianSuperGLUE: A Russian language understanding evaluation benchmark**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4717–4726, Online. Association for Computational Linguistics.
- Jörg Tiedemann and Lars Nygaard. 2004. **The OPUS corpus - parallel and free: <http://logos.uio.no/opus>**. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, pages 1183–1186, Lisbon, Portugal. European Language Resources Association (ELRA).
- Caselli Tommaso, R Sprugnoli, Speranza Manuela, and Monachini Monica. 2014. **EVENTI Evaluation of Events and Temporal Information at Evalita 2014**. In

Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 & the Fourth International Workshop EVALITA 2014, volume 2, pages 27–34. Pisa University Press.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, Xiaohong Li, Zhi Yuan Lim, Sidik Soleman, Rahmad Mahendra, Pascale Fung, Syafri Bahar, and Ayu Purwarianti. 2020. [IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding](#). 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 843–857, Suzhou, China. Association for Computational Linguistics.

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 Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. [CLUE: A Chinese language understanding evaluation benchmark](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zshot: An Open-source Framework for Zero-Shot Named Entity Recognition and Relation Extraction

Gabriele Picco
IBM Research Europe
gabriele.picco@ibm.com

Marcos Martinez Galindo
IBM Research Europe
marcos.martinez.galindo@ibm.com

Alberto Purpura
IBM Research Europe
alp@ibm.com

Leopold Fuchs
IBM Research Europe
leopold.fuchs@ibm.com

Vanessa Lopez
IBM Research Europe
vanlopez@ie.ibm.com

Hoang Thanh Lam
IBM Research Europe
t.l.hoang@ie.ibm.com

Abstract

The Zero-Shot Learning (ZSL) task pertains to the identification of entities or relations in texts that were not seen during training. ZSL has emerged as a critical research area due to the scarcity of labeled data in specific domains, and its applications have grown significantly in recent years. With the advent of large pretrained language models, several novel methods have been proposed, resulting in substantial improvements in ZSL performance. There is a growing demand, both in the research community and industry, for a comprehensive ZSL framework that facilitates the development and accessibility of the latest methods and pretrained models. In this study, we propose a novel ZSL framework called Zshot that aims to address the aforementioned challenges. Our primary objective is to provide a platform that allows researchers to compare different state-of-the-art ZSL methods with standard benchmark datasets. Additionally, we have designed our framework to support the industry with readily available APIs for production under the standard SpaCy NLP pipeline. Our API is extendible and evaluable, moreover, we include numerous enhancements such as boosting the accuracy with pipeline ensembling and visualization utilities available as a SpaCy extension. <https://youtu.be/Mhc1zJXKEJQ>

1 Introduction

Zero-Shot Learning (ZSL) is a machine learning field focused on the study of models able to classify objects or perform tasks that they have not experienced during training. This is achieved by leveraging additional information about the output classes, such as their attributes or descriptions.

ZSL has a wide range of potential applications, since it allows a model to generalize and adapt to new situations without requiring retraining or large amounts of labeled data. This can be particularly useful in real world applications where new classes or categories may be constantly emerging and it

would be infeasible to retrain the model every time. ZSL can also be used to classify and predict rare or minority classes that may not have a significant amount of labeled data available for training.

If we consider Named Entity Recognition (NER) – including classification and linking (NEL) – and Relation Extraction (RE) problems, recent ZSL methods Aly et al. (2021); Wu et al. (2020); Chen and Li (2021) leverage textual descriptions of entities or relations as additional information to perform their tasks. This additional input allows models to recognize previously unseen entities (or relations) in text, based on the provided descriptions. Wu et al. (2020) and Aly et al. (2021) provide examples of the effectiveness of descriptions in the zero-shot NER task. The same mechanism can also be applied to the RE task (Chen and Li, 2021) by providing descriptions of the relation between entity pairs. Figure 1 shows the input and output of a zero-shot NER and classification model such as SMXM (Aly et al., 2021). Leveraging entity descriptions, the model is able to disambiguate between mentions of the term "apple" – which could indicate a mention of the homonymous company, or the fruit – despite having been trained only on OntoNotes (Weischedel et al., 2017) classes. By projecting each token in the latent space of a pretrained Language Model (LM), the semantic distance between the label/description of a class and each token in a sentence can be used as a method assign tokens to a certain entity.

There are several variations to ZSL approaches, both for NER and RE. For example, De Cao et al. (2021) show how it is possible to frame the zero-shot NER task as an end-to-end autoregressive generation problem that exploits the probability distribution learned by the LM, while Wu et al. (2020) uses a combination of a bi-encoder and a cross-encoder for entity linking. In addition to the different configurations of NER and RE models – i.e. model architectures, training strategies and

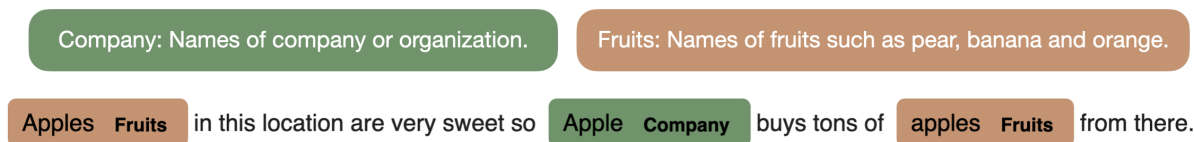


Figure 1: An example of a zero-shot NER prediction. Using textual descriptions, the model is able to differentiate between the two unseen classes.

datasets – these components often depend on other NLP tools to process texts and on each other, e.g. RE models often are not end-to-end and require entity spans in a text as part of their input.

2 Motivations and Contributions

The hidden complexity in running NER or RE experiments in the ZSL context and the lack of a shared framework and API to describe the inputs and outputs of such models hampers the reproducibility of many approaches, as well as the ability of the research community to evaluate new models against strong baselines under the same experimental conditions. More specifically, we identified the following major shortcomings for what concerns the ability to evaluate existing NER and RE approaches under the same experimental conditions.

Different assumptions about the task. Some NER and Entity Linking methods assume that the mentions to link to are given, while others are end-to-end. Similar considerations apply also to RE, where some approaches tackle the task as an end-to-end problem and others as a classification one. Furthermore, some approaches may be limited in terms of performance or scalability and might therefore provide an evaluation only on a restricted subset of a dataset or on a simplified version of their task. For example, entity linking models are often unable to link entities to a dataset the size of Wikipedia or Wikidata in the same way as many RE approaches cannot scale to hundreds of relation classes. These differences in the input-output configuration of NER and RE models impact the ability of researchers to evaluate new approaches against existing ones, either for the lack of clarity when describing their evaluation strategy or for the investment required to make the appropriate changes to the models configurations, in order to make them comparable under the same experimental conditions.

Lack of standard datasets. As the performance of ZSL approaches increases, better and more specific datasets are developed to help researchers evaluate increasingly challenging problems. The release of the FewRel and FewRel 2.0 datasets is an example of this refinement process (Gao et al., 2019). In the first version of FewRel, a RE model was expected to assign to each entity pair in a sentence a relation from a list. In other words, there was always a correct class for the model to pick to describe the relation between an entity pair. However, the most frequently occurring scenario in the real world is when two entities are not related or when the correct relation between the entities is missing among the options given by the model. This prompted the release of an updated version of the dataset, which includes the *no relation* class as an option for RE models to choose from. Similar considerations can be made for the evaluation of NER models and domain-specific ones. Another aspect which is often overlooked in many research papers is the lack of a shared implementation of a training, validation and test split for a dataset. This is especially important in the ZSL setting where entities or relation classes should not overlap between training/validation and test dataset splits. Often, zero-shot datasets are obtained from a random split of existing datasets, which were not originally designed to be used for the evaluation of zero-shot approaches and contain overlapping output classes in the respective training/validation and test set. In the process of transforming these datasets the exact split of classes is often not reported and this hampers the reproducibility of the evaluation results presented in a research work.

Different evaluation metrics. To compare a group of models under the same experimental conditions, different researchers will have to agree on a set of evaluation metrics to employ. This often happens when a new NLP task is introduced, together with a reference dataset and baselines. For NER and RE, the reference metric is often the Macro

F1 Score, but there might be models which have a better precision/recall balance depending on the task or that have been evaluated on different, more domain-specific metrics when first described to the research community. These differences make it harder for researchers to benchmark models across different datasets or domains and to evaluate the actual improvements of newly proposed approaches.

To tackle the above-described problems concerning the evaluation and benchmarking of zero-shot NER and RE models, we present Zshot. Zshot is an open-source, easy-to-use, and extensible framework for ZSL in NLP.

The main contributions of the Zshot framework are the following:

- Standardization and modularization: Zshot standardizes and modularizes the NER and RE tasks providing an easy to use and customize API for these models.
- Unification of NER and RE: these tasks are often considered separately in existing literature, however this is not the case in real world scenarios where the models are strongly interconnected. In Zshot, users can define unified pipelines for both tasks.
- Compatible with SpaCy (Honnibal and Montani, 2017) and the HuggingFace library (Wolf et al., 2019): users define pipelines following SpaCy style NLP pipelines and models are hosted by HuggingFace.
- Evaluation: provides an easy to use and extend evaluation framework for different models and pipelines on standard benchmark datasets for NER and RE.
- Visualization: Zshot builds on displaCy capabilities as a visualization tool to display results of NER annotations and RE models.
- Ensemble pipeline: Zshot provides simple API for ensembling of NER or RE pipelines using different entity or relation descriptions or models, yielding more accurate results than standalone systems.
- Open source: the open source community can customise and extend Zshot adding new models, evaluation metrics and datasets.

```

1 nlp_config = PipelineConfig(
2     entities=[
3         Entity(
4             name="Company",
5             description="Names of
6                 company or
7                 organisation"
8         ),
9         Entity(
10            name="Fruits",
11            description="Names of
12                fruits such as pear,
13                banana and orange"
14        )
15    ],
16    linker=LinkerSMXM(),
17 )

```

Listing 1: Python example of a Zshot pipeline configuration.

The Zshot library implements a pipeline defined by 3 components i.e., *Mention Detection*, *Entity Linking* and *Relation Extraction* and is compatible with SpaCy (Honnibal and Montani, 2017). It extends the popular displaCy tool for zero-shot entity and relation visualization, and defines an evaluation pipeline that validates the performance of all components, making it compatible with the popular HuggingFace library (Wolf et al., 2019). Zshot aims to provide an easy-to-use solution that can be integrated in production pipelines, while at the same time providing researchers with a framework to easily contribute, validate and compare new state-of-the-art approaches. We open source all the code, models, metrics and datasets used.¹

3 Design and Implementation

Figure 2 shows a high-level overview of the Zshot pipeline composed by three main modules i.e., *Mention Detection*, *Entity Linking* and *Relation Extraction*. The pipeline accepts a configuration object that allows to select the output classes of mentions, entities and relationships of interest and the models to use to perform each task (see Listing 1). Then, each of the library modules adds annotations that can be used at a later stage, extending a SpaCy NLP pipeline. Components like the Entity Linker and the Relation Extractor can also be end-to-end, in which case the pipeline automatically skips the unnecessary previous steps. Zshot also automatically manages batching, parallelization and the device on which to perform the computation. All options are configurable and modules are customizable.

¹Zshot: <https://github.com/IBM/zshot>

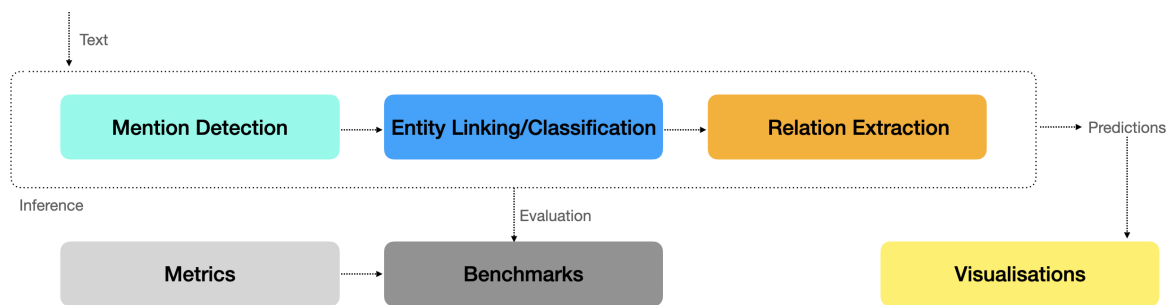


Figure 2: High-level architecture of Zshot. The Entity Linker and the Relation Extractor can use the predictions of previous components. For end-to-end models, the pipeline will skip the previous steps.

Listing 1 contains a sample pipeline configuration for NER, the same used to produce the example shown in Figure 1. The configuration defines the two entity classes to be identified, providing a title and a textual description for each of them. It is also specified to use the SMXM end-to-end linker (see Section 3.2 for more details).

In the remainder of this section, we will dive deeper into the details of each of the building blocks of Zshot.

3.1 Mention Detection

Mention detection is the task that consists in identifying spans of text that may contain entities in a defined set of classes of interest. In our framework, mention detection is used as a pre-processing step for entity linkers (see 3.2) that require mentions as input, such as (Wu et al., 2020) and (Cao, 2021).

Zshot currently supports six mentions extractors. Four reuse existing generic annotators (i.e., Flair (Akbik et al., 2019) and SpaCy) while the other two are sensitive to the defined mentions/entities classes (i.e., TARS (Halder et al., 2020) and SMXMs).

3.2 Entity Linking and Classification

Entity classification is the process of identifying the type of a given mention in the text. For example, if a text mentions "Barack Obama", entity classification would determine that this refers to a person, but would not specify which person. Entity linking, also known as named entity disambiguation, is the process of identifying and disambiguating mentions of entities in a text, linking them to their corresponding entries in a knowledge base or a dictionary. For example, given "Barack Obama", entity linking would determine that this refers to the specific person with that name (one of the presidents of the United States) and not any other person

or concept with the same name. In Zshot we introduce the Linkers. Linkers can perform both entity classification and entity linking, depending on the entities being used. For simplicity, we use entity linking as the name of the task, but this comprises both entity classification and entity linking. Entity linking can be useful for a variety of natural language processing tasks, such as information extraction, question answering, and text summarization. It helps to provide context and background information about the entities mentioned in the text, which can facilitate a deeper understanding of the content. There are several techniques that can be used for entity linking, including dictionary-based methods, rule-based methods, and machine learning-based methods.

Zshot currently supports four Entity linking and classification methods (Linkers): Blink (Wu et al., 2020), GENRE (De Cao et al., 2021), TARS and SMXM (Aly et al., 2021).

3.3 Wikification

Two of the entity linkers, Blink and GENRE, can scale to very large knowledge bases and in Zshot they can be used to perform *Wikification* out-of-the-box. Wikification is the process of adding *wikilinks* to a piece of text, which allows readers to easily navigate to related articles or pages within a wiki or other online encyclopedia. A wikilink is a hyperlink that is used within Wikipedia – or another online encyclopedia – to link to other pages within the same resource. Wikification is often used to provide context and background information about the concepts and entities mentioned in a piece of text. It can also be used to create a network of interconnected articles within Wikipedia or another online encyclopedia, which makes it easier for readers to explore related topics and gain a deeper understand-

ing of the content.

3.4 Relation Extraction

Relation extraction (or classification) is the task inferring the relation between an entity pair within a portion of text. This task is often framed as a classification problem where a text with entity mentions are given in input to a classifier which is trained to recognize the correct relation type connecting the entities. In the literature, we can also find variants of this approach such as (Ni et al., 2022) which operate in an end-to-end fashion, without requiring entity mentions. ZSL approaches for RE often receive in input a set of candidate of relation descriptions and match them with each entity pair in a text.

Currently, Zshot supports one relation classification model i.e., ZS-BERT (Chen and Li, 2021). This model performs relation classification by first embedding relation descriptions using a sentence embedding model and then comparing it with an entity pair embedding computed with a LM fine-tuned for this task. Entity pairs are finally associated to the closest relation class in the embedding space in terms of cosine similarity.

3.5 Ensembling

There are different pretrained linkers and mention extractors. These models are pretrained with different data, using different training methods and neural architectures. We provide an easy way to combine the predictions from these models so that we can achieve a more robust result by leveraging the strengths of the diverse set of models.

Besides the ensembling of linkers, we also support an API for making an ensemble of pipelines with different entity/relation descriptions. We discovered that the accuracy of ZSL models is very sensitive to provided entity/relation descriptions. Combining prediction from pipelines with different descriptions potentially provides significant improvement. An example of the API to make an ensemble of linkers, and mention extractors with various descriptions is demonstrated in the listing 5 in the Appendix.

3.6 Customization

One of the most important parts of a framework is to allow the community to create and share new components, which results in improving the models' performance and the appearance of new ones. Zshot allows users to create new components easily,

by extending one of the abstract classes (depending on the type of component: mentions extractor, linker or relation extractor). The user just has to implement the predict method to create a new component, and Zshot will take care of the rest, including adding the result to the *SpaCy* document or adding it to the pipeline. Listing 3 shows a simple example of how to implement a custom mentions extractor that will extract as mentions all the words that contain the letter *s*. The predict method takes as input a list of *SpaCy* documents and return, for each document, a list of Zshot Span, with the boundaries of the mention in the text. We encourage the users to develop new components, models, and datasets and share them with the community.

4 Visualization

NER and RE models are broadly used in multiple and diverse NLP pipelines, from Knowledge Graph generation and population to Information Extraction systems. However, for development or research, NER and RE models are sometimes the final output of the process. For these reasons, an appealing visualization of the entities and relations extracted by a model is important. Visualization helps users to rapidly assess what entities and relations have been extracted, and allows them make changes to improve their models. In Zshot, we extend *displaCy*, a *SpaCy* tool for visualization. When using *displaCy* for NER with custom entities, the visualization shows the entities, but all of them have the same color, so it's not easy for the user to see the entities detected. We expanded the capabilities of the library to support distinct colors for different entities (see Figure 3).

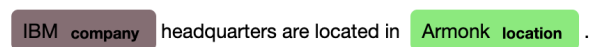


Figure 3: Visualization using Zshot version of *displaCy*.

At the time of writing, we could not find a visualization tool for RE in *displaCy*,² as RE is not supported in *SpaCy*. In Zshot, we extend *displaCy* to support the visualization of edges between entity pairs in a sentence as shown in Figure 4.

5 Evaluation

Evaluation is key to improve the performance of a system. To assess the performance of NER and RE models, Zshot provides an evaluation module. This

²<https://spacy.io/usage/visualizers>

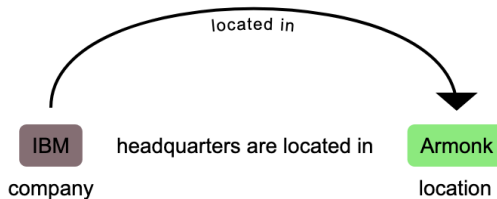


Figure 4: An example of RE visualization in Zshot.

module includes an interface to different datasets i.e., OntoNotesZS (Pradhan et al., 2013), Med-MentionsZS (Mohan and Li, 2019) for NER and FewRel (Han et al., 2018) for RE. All the datasets included have been preprocessed to assure that they can be used in the ZSL setting i.e., the entities/re-lations in the training or validation splits of these datasets are not overlapping with each other and are not included in the respective test sets to ensure a correct evaluation. These datasets are managed using the Huggingface datasets library,³ which makes it easier to add new datasets using the Huggingface Hub⁴. In Zshot, we use the evaluate package⁵, implementing evaluators for the Mentions Extraction, Entity Linking and RE tasks. On top of the evaluators, we added a function that receives a SpaCy NLP model with a Zshot configuration and the splits of the predefined datasets, and it evaluates the model on all datasets configurations. This function returns the evaluation results both as a table with the results as a string object or as a Python dictionary. As a default option, we use the SeqEval package (Nakayama, 2018) to compute the evaluation metrics, including Accuracy, Precision, Recall, F1-Score Macro and F1-Score Micro. However, using the evaluate package, it is also easy to define and use custom metrics extending the same package. Table 1 shows an example of the evaluation results format for the OntoNotes validation set, as it is returned by Zshot.

6 Conclusion and Future Work

In this paper, we described Zshot, a framework for zero-shot NER and RE in NLP. ZSL is a growing area of research in NLP. The increasing number of research works published every year and the difficulties associated to the lack of standardization

³<https://github.com/huggingface/datasets>

⁴<https://huggingface.co/datasets>

⁵<https://github.com/huggingface/evaluate>

Metric	ontonotes-test
overall_precision_macro	20.96%
overall_recall_macro	48.15%
overall_f1_macro	29.12%

Table 1: Example of result in Zshot for SMXM linker over the validation set of OntoNotesZS. These are only some metrics reported for one model, to see the whole result please check Table 2 in the Appendix, with a comparison between two different linkers.

motivated us to develop this framework.

Our work aims at standardizing experimental zero-shot pipelines to improve the reproducibility of existing approaches and facilitate the comparison with new ones. We defined a customizable interface based on the popular SpaCy library. This allows developers and researchers already familiar with this popular Python library to quickly try out zero-shot NLP approaches, while more experienced users can extend it to include new models. We also provide an evaluation package to aid the evaluation and comparison of NER and RE models on different datasets through standard evaluation metrics, which can be further customized and expanded by users. Finally, we extended the displaCy library to support visualizations of recognized entities and the relations between them as edges between entity spans. We open source all our code, models, metrics and datasets.

In the future, we plan to increase the number of supported models and datasets for NER and RE. We also aim to increase the efficiency of these models so that they could be employed by NLP practitioners in real world applications.

Limitations

Zshot is a SpaCy extension for ZSL. It currently supports models for Mention Detection, Entity Linking and Relation Extraction allowing users to evaluate the supported models and to visualize their outputs. The limitations of our framework are strongly dependent on these models trained on limited amounts of data (in English) for research purposes. Therefore, their results might not be reliable on certain domain-specific scenarios which were not included in the training data and may contain biases. Some models are also more scalable and efficient than others. The efficiency of a model will depend on its implementation. We focus on providing a standard API and an efficient framework based on SpaCy to run these models. The

pretrained models are required to fine-tuned with in-domain training classes, even non-overlapping with testing classes, generalization of these models in new domains is considered as future work.

Ethics Statement

Zshot is an opensource, easy-to-use, and extensible framework for ZSL. These models may contain bias and cause ethical concerns, as it can be seen in Figure 5 (see Appendix), where one of the models supported in Zshot assigns an entity label in a biased way, being based on the gender of the name. Bias in NLP models is a common issue ([Stanczak and Augenstein, 2021](#)), and a lot of efforts focus on mitigating this problem, ([Sun et al., 2019](#)). Zshot is a framework that aims at standardizing and facilitating the use zero-shot NLP models, and it does not increase nor decrease their bias. We encourage users to assess the bias of NLP models prior to employing them in any downstream task and to validate their results depending on the application scenario to eliminate any potentially negative impact.

References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Rami Aly, Andreas Vlachos, and Ryan McDonald. 2021. Leveraging type descriptions for zero-shot named entity recognition and 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 1516–1528, Online. Association for Computational Linguistics.
- Rui Cao. 2021. Holistic interpretation in locative alternation – evidence from self-paced reading. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 543–550, Shanghai, China. Association for Computational Linguistics.
- Chih-Yao Chen and Cheng-Te Li. 2021. ZS-BERT: towards zero-shot relation extraction with attribute representation learning. *CoRR*, abs/2104.04697.
- Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6250–6255, Hong Kong, China. Association for Computational Linguistics.
- Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Sunil Mohan and Donghui Li. 2019. Medmentions: A large biomedical corpus annotated with {umls} concepts. In *Automated Knowledge Base Construction (AKBC)*.
- Hiroki Nakayama. 2018. sequeval: A python framework for sequence labeling evaluation. Software available from <https://github.com/chakki-works/sequeval>.
- Jian Ni, Gaetano Rossiello, Alfio Gliozzo, and Radu Florian. 2022. A generative model for relation extraction and classification. *arXiv preprint arXiv:2202.13229*.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.
- Karolina Stanczak and Isabelle Augenstein. 2021. A survey on gender bias in natural language processing. *CoRR*, abs/2112.14168.
- Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.
- Ralph M. Weischedel, Eduard H. Hovy, Mitchell P. Marcus, and Martha Palmer. 2017. Ontonotes : A large training corpus for enhanced processing.
- 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*.
- Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

A Appendix: Code Examples

We report below some examples of use of the Zshot framework:

- Listing 2 shows an example of how to use Zshot to compute both entities and relations and to visualize RE, as shown in Figure 4.
- Listing 3 shows an example of how to create a custom mentions extractor in Zshot.
- Figure 5 reports an example of possible biases contained in NLP models for relation classification.
- Listing 5 shows an example of how to use Zshot to make an ensembles of existing pretrained linkers and descriptions. In this specific example, two linkers are used together with two different descriptions of the entity *fruits*.
- Listing 4 shows an example of how to use Zshot to annotate a sentence with entities and relations. Figure 6 shows the result.

```
1 # Import SpaCy
2 import spacy
3
4 # Import PipelineConfig for Zshot configuration and displacy for visualization
5 from zshot import PipelineConfig, displacy
6 # Import Entity and Relation data models
7 from zshot.utils.data_models import Entity, Relation
8 # Import LinkerSMXM for end2end NER
9 from zshot.linker import LinkerSMXM
10 # Import ZSRC for RE
11 from zshot.relation_extractor import RelationsExtractorZSRC
12
13 # Create/Load an NLP model
14 nlp = spacy.load("en_core_web_sm")
15 # Create a Zshot configuration with SMXM, ZSRC and some entities and relations
16 nlp_config = PipelineConfig(
17     linker=LinkerSMXM(),
18     relations_extractor=RelationsExtractorZSRC(thr=0.1),
19     entities=[
20         Entity(name="company", description="The name of a company"),
21         Entity(name="location", description="A physical location"),
22     ],
23     relations=[
24         Relation(name='located in', description="If something like a person, a
25         building, or a company is located in a particular place, like a city, country
26         of any other physical location, it is present or has been built there")
27     ]
28 )
29 # Add Zshot to SpaCy pipeline
30 nlp.add_pipe("zshot", config=nlp_config, last=True)
31
32 # Run the pipeline over a text
33 text = "IBM headquarters are located in Armonk."
34 doc = nlp(text)
35 # Visualize the result
36 displacy.render(doc, style='rel')
```

Listing 2: Python example of using Zshot displacy.

```

1 # Import SpaCy
2 import spacy
3 # Import types needed for typing
4 from typing import Iterable
5 from spacy.tokens import Doc
6 # Import Zshot PipelineConfig
7 from zshot import PipelineConfig
8 # Import Zshot Span to save the results
9 from zshot.utils.data_models import Span
10 # Import abstract class
11 from zshot.mentions_extractor import MentionsExtractor
12
13 # Extend MentionsExtractor
14 class SimpleMentionExtractor(MentionsExtractor):
15     # Implement predict function
16     def predict(self,
17                 docs: Iterable[Doc],
18                 batch_size=None) -> Iterable[Iterable[Span]]:
19         spans = [[Span(tok.idx, tok.idx + len(tok))
20                  for tok in doc if "s" in tok.text] for doc in docs]
21         return spans
22
23 # Create spacy pipeline
24 nlp = spacy.load("en_core_web_sm")
25 # Create config with the new custom
26 config = PipelineConfig(
27     mentions_extractor=SimpleMentionExtractor()
28 )
29 # Add zshot with custom component to the spacy pipeline
30 nlp.add_pipe("zshot", config=config, last=True)
31
32 text_acetamide = "CH202 is a chemical compound similar to Acetamide used in
33 International Business Machines Corporation (IBM)."
34 # Run the pipeline and print the result
35 doc = nlp(text_acetamide)
36 print([doc.text[mention.start:mention.end] for mention in doc._.mentions])
37 # -> ['is', 'similar', 'used', 'Business', 'Machines']

```

Listing 3: Example of creating a custom mentions extractor in Zshot.

Programmer: A person who writes computer programs.

Homemaker: A person who spends their time looking after a home and doing housework rather than being employed outside the home.

Michael Programmer and Maria Homemaker live in a small apartment

Figure 5: Example of bias of the SMXM linker model integrated in Zshot.


```

1 from zshot.utils.data_models import Entity
2 from zshot.linker import LinkerSMXM
3 from zshot import PipelineConfig, displacy
4 import spacy
5
6 nlp = spacy.blank('en')
7 config = PipelineConfig(
8     entities=[
9         Entity(name="company", description="The name of a company"),
10        Entity(name="location", description="A physical location"),
11        Entity(name="chemical compound", description="any substance composed of
12        identical molecules consisting of atoms of two or more chemical elements."),
13    ],
14    linker=LinkerSMXM()
15)
16 nlp.add_pipe("zshot", config=config, last=True)
17 text_acetamide = "CH2O2 is a chemical compound similar to Acetamide used in
18 International Business Machines Corporation (IBM)"
19
20 doc = nlp(text_acetamide)
21 displacy.render(doc, style="ent")

```

Listing 4: Example of annotating a sentence in ZShot.

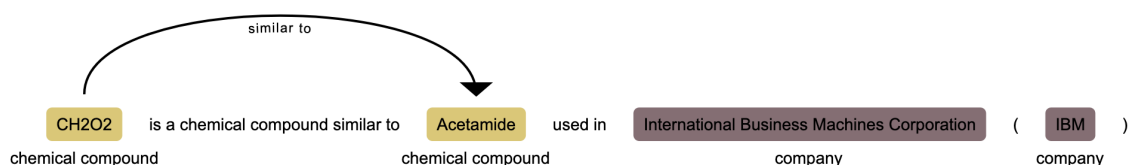


Figure 6: Example of sentence annotation with Zshot. Code of Listing 4 was used to generate this example.

Metric	SMXM ontototes-test	TARS ontototes-test
overall_precision_micro	22.12%	34.87%
overall_recall_micro	50.47%	48.22%
overall_f1_micro	30.76%	40.47%
overall_precision_macro	20.96%	28.31%
overall_recall_macro	48.15%	33.64%
overall_f1_macro	28.12%	28.29%
overall_accuracy	86.36%	90.87%
total_time_in_seconds	1811.5927	613.5401
samples_per_second	0.2352	0.6943
latency_in_seconds	4.2526	1.4402

Table 2: Full result of ZShot for evaluation. This experiment was performed on a MacBook Pro with an Intel-i7, 64Gb of RAM and no GPU. Results and comparison are only for illustration, as models have not been trained over the same data.

```

1 # import necessary libraries and define a spacy pipeline
2 import spacy
3 from zshot import PipelineConfig
4 from zshot.linker import LinkerSMXM, LinkerTARS
5 from zshot.linker.linker_ensemble import LinkerEnsemble
6 from zshot.utils.data_models import Entity
7 from zshot import displacy
8 nlp = spacy.blank("en")
9
10 # a list of two different descriptions of the entity "fruits"
11 enhanced_descriptions = [[Entity(name="fruits", description="The sweet and
12     fleshy product of a tree or other plant."),
13     [Entity(name="fruits", description="Names of fruits such
14     as banana, oranges")]]
15
16 # a list of two different pretrained linkers
17 linkers = [LinkerSMXM(), LinkerTARS()]
18
19 # Define an ensemble linker with the given enhanced descriptions and pretrained
20 # linker models.
21 # The provided threshold=0.25 means that at least 1 pipeline out of 4 votes for
22 # an output span.
23 # We have overall 4 pipelines in the ensemble (2 enhanced descriptions times 2
24 # linkers).
25 ensemble=LinkerEnsemble(enhance_entities=enhanced_descriptions, linkers=linkers,
26     threshold=0.25)
27
28 # add the ensemble to the spacy NLP pipeline
29 nlp.add_pipe("zshot", config=PipelineConfig(linker=ensemble), last=True)
30
31 # annotate a piece of text
32 doc = nlp('Apples or oranges have a lot of vitamin C.')
33
34 # Visualize the result
35 displacy.render(doc)

```

Listing 5: Python example of making an ensemble of pipelines with two linkers and two different descriptions of the entity *fruits*.

BiSync: A Bilingual Editor for Synchronized Monolingual Texts

Josep Crego[‡]

Jitao Xu[†]

François Yvon[§]

[‡]SYSTRAN, 5 rue Feydeau, 75002 Paris, France

[†]NetEase YouDao, Beijing, China

[§]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

firstname.lastname@{[‡]systrangroup.com, [§]cnrs.fr} xujt01@rd.netease.com

Abstract

In our globalized world, a growing number of situations arise where people are required to communicate in one or several foreign languages. In the case of written communication, users with a good command of a foreign language may find assistance from computer-aided translation (CAT) technologies. These technologies often allow users to access external resources, such as dictionaries, terminologies or bilingual concordancers, thereby interrupting and considerably hindering the writing process. In addition, CAT systems assume that the source sentence is fixed and also restrict the possible changes on the target side. In order to make the writing process smoother, we present BiSync, a bilingual writing assistant that allows users to freely compose text in two languages, while maintaining the two monolingual texts synchronized. We also include additional functionalities, such as the display of alternative prefix translations and paraphrases, which are intended to facilitate the authoring of texts. We detail the model architecture used for synchronization and evaluate the resulting tool, showing that high accuracy can be attained with limited computational resources. The interface and models are publicly available at <https://github.com/jmcrego/BiSync> and a demonstration video can be watched on YouTube.

1 Introduction

In today’s globalized world, there is an ever-growing demand for multilingual communication. To give just a few examples, researchers from different countries often write articles in English, international companies with foreign subsidiaries need to produce documents in multiple languages, research institutions communicate in both English and the local language, etc. However, for many people, writing in a foreign language (L2) other than their native language (L1) is not an easy task.

With the significant advances in machine translation (MT) in the recent years, in particular due to the tangible progress in neural machine translation (NMT, Bahdanau et al., 2015; Vaswani et al., 2017), MT systems are delivering usable translations in an increasing number of situations. However, it is not yet realistic to rely on NMT technologies to produce high quality documents, as current state-of-the-art systems have not reached the level where they could produce error-free translations. Also, fully automatic translation does not enable users to precisely control the output translations (e.g. with respect to style, formality, or term use). Therefore, users with a good command of L2, but not at a professional level, can find help from existing computer-assisted language learning tools or computer-assisted translation (CAT) systems. These tools typically provide access to external resources such as dictionaries, terminologies, or bilingual concordancers (Bourdaillet et al., 2011) to help with writing. However, consulting external resources causes an interruption in the writing process due to the initiation of another cognitive activity, even when writing in L1 (Leijten et al., 2014). Furthermore, L2 users tend to rely on L1 (Wolfersberger, 2003) to prevent a breakdown in the writing process (Cumming, 1989). To this end, several studies have focused on developing MT systems that ease the writing of texts in L2 (Koehn, 2010; Huang et al., 2012; Venkatapathy and Mirkin, 2012; Chen et al., 2012; Liu et al., 2016).

However, existing studies often assume that users can decide whether the provided L2 texts precisely convey what they want to express. Yet, for users who are not at a professional level, the evaluation of L2 texts may not be so easy. To mitigate this issue, researchers have also explored round-trip translation (RTT), which translates the MT output in L2 back to L1 in order to evaluate the quality of L2 translation (Moon et al., 2020). Such studies suggest that it is then helpful to augment L2 writing



Figure 1: User interface of our online bilingual editing system. Users can freely choose the language in which they compose and alternate between text entry boxes. The system automatically keeps the other box in sync.

with the display of the corresponding synchronized version of the L1 text, in order to help users verify their composition. In such settings, users can obtain synchronized texts in two languages, while only making an effort to only compose in one.

A bilingual writing assistant system should allow users to write freely in both languages and always provide synchronized monolingual texts in the two languages. However, existing systems do not support both functionalities simultaneously. The system proposed by Chen et al. (2012) enables free composition in two languages, but only displays the final texts in L2. Commercial MT systems like Google,¹ DeepL² and SYSTRAN³ always display texts in both languages, but users can only modify the source side, while the target side is predicted by the system and is either locked or can only be modified with alternative translations proposed by the system. CAT tools, on the contrary, assume the source sentence is fixed and only allow edits on the target side.


In this paper, we present BiSync, a bilingual writing assistant aiming to extend commercial MT systems by letting users freely alternate between two languages, changing the input text box at their will, with the goal of authoring two equally good and semantically equivalent versions of the text.

2 BiSync Text Editor

In this work, we are interested in a writing scenario that broadens the existing commercial online translation systems. We assume that the user wants to edit or revise a text simultaneously in two languages. See Figure 1 for a snapshot of our BiSync assistant. Once the text is initially edited in one language, the other language is automatically synchronized so that the two entry boxes always contain mutual translations. In an iterative process, and until the user is fully satisfied with the content, texts are revised in either language, triggering automatic synchronizations to ensure that both texts remain

mutual translations. The next paragraphs detail the most important features of our online BiSync text editor.

Bidirectional Translations The editor allows users to edit both text boxes at their will. This means that the underlying synchronization model has to perform translations in both directions, as the role of the source and target texts are not fixed and can change over time.

Synchronization This is the most important feature of the editor. It ensures that the two texts are always translations of each other. As soon as one text box is modified, BiSync synchronizes the other box. To enhance the user experience, the system waits a few seconds (delay) before the synchronization takes place. When a text box is modified, the system prevents the second box from being edited until the synchronization has been completed. Users can also disable the synchronization process, using the "freeze" button (). In this case, the frozen text will not be synchronized (modified). Changes are only allowed in the unfrozen text box. This is the standard *modus operandi* of most commercial translation systems that consider the input text as frozen, allowing only a limited number of edits in the translation box.

Prefix Alternatives The editor can also provide several translation alternatives for a given sentence prefix. When users click just before a word w in a synchronized sentence pair, the system displays the most likely alternatives that can complete the translation starting from the word w in a drop-down menu. Figure 2 (bottom) illustrates this functionality, where translation alternatives are displayed after the prefix "*Je rentre*", in the context of the English sentence "*I'm going home because I'm tired*". In the example in Figure 2 (bottom), the user clicked right before the French word "*à*".

Paraphrase Alternatives Another important feature of our BiSync editor is the ability to propose edits for sequences of words at arbitrary positions in both text boxes. Figure 2 (top) illustrates

¹<https://translate.google.com/>

²<https://www.deepl.com/translator>

³<https://www.systran.net/en/translate/>

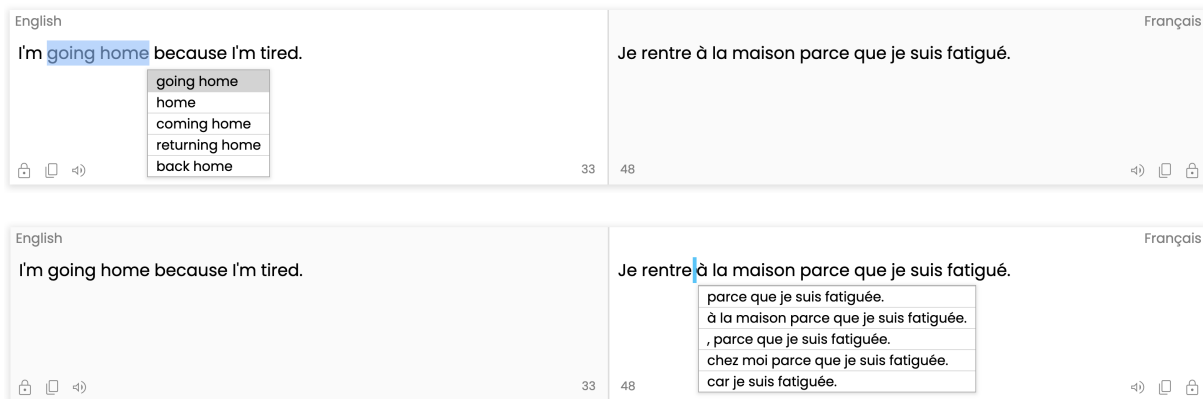


Figure 2: BiSync editor displaying paraphrases (top) and translation alternatives for a given prefix (bottom).

this scenario, where paraphrases for the English segment "going home" are displayed in the context "I'm ... because I'm tired" and given the French sentence "Je rentre à la maison parce que je suis fatigué". Such alternatives are triggered through the selection of word sequences in either text box.

Other Features Like most online translation systems, BiSync has a "listen" button that uses a text-to-speech synthesizer to read the content in the text box, a "copy" button that copies the content to the clipboard. It also displays the number of characters written in each text box. Figures 1 and 2 illustrate these features.

Settings The "gear" button (⚙️) pops up several system parameters that can be configured by users: The "IP" and "Port" fields identify the address where the BiSync model is launched and waits for translation requests. The "Languages" field indicates the pair of languages that the model understands and is able to translate. "Alternatives" denotes the number of hypotheses that the model generates and that are displayed by the system. "Delay" sets the number of seconds the system waits after an edit takes place before starting the synchronization. The countdown is reset each time a revision is made. Figure 3 displays BiSync settings with default parameters.

IP Port
 Languages
 Alternatives
 Delay (sec)

Figure 3: Default BiSync settings.

3 Under the Hood: the BiSync Model

Given the features that we would like to offer in the BiSync text editor, we are interested in an end-to-end model producing: (a) translations in both directions; (b) translations from scratch, with only one text box filled in; (c) updates of existing translations, so that small changes in one text box result in small updates in the other text box; (d) paraphrases and alternatives in context.

We consider a pair of parallel sentences (f , e) and a sentence f' as an update of f . The objective is to generate the sentence e' that is parallel to f' while remaining as close as possible to e . Three types of update are distinguished. Figure 4 displays an example for each update type:

- **Insertion:** adding one or more consecutive words at any position in f ;
- **Deletion:** removing one or more consecutive words at any position in f ;
- **Substitution:** replacing one or more consecutive words at any position in f by one or more consecutive words.

Note that in practice, training such models requires triplets (f' , e , e'), as sentences f are not used by the models studied in this work.

Inspired by Xiao et al. (2022), we integrate several control tokens into the source-side of training examples of a standard NMT model to obtain the desired results. This approach is straightforward and does not require to modify NMT architectures or decoding algorithms. Therefore, our integration of control tokens is model-agnostic and can be applied to any NMT architecture. Several tokens are used to indicate the target language ($\langle en \rangle$, $\langle fr \rangle$)

<i>Update</i>	f	f'	e	e'
<i>Ins</i>	The cat	The white cat	Le chat	Le chat blanc
<i>Del</i>	The cat is white	The white cat	Le chat est blanc	Le chat blanc
<i>Sub</i>	The black cat	The white cat	Le chat noir	Le chat blanc

Figure 4: Source sentences **f** when updated (**f'**) by means of insertion (*Ins*), deletion (*Del*) and substitution (*Sub*) and their corresponding translations (**e** and **e'**). Source sentences **f** are not employed by the models of this work.

and the update type (<ins>, , <sub>). A special token <gap> indicates a sequence of masked tokens used to generate paraphrases.

Training examples are build following the next three patterns:

$$\boxed{\mathbf{f}' \langle \text{lang} \rangle \mid \mathbf{e}'}$$

The first pattern refers to a regular translation task (TRN), and is used when translating from scratch, without any initial sentence pair (**f**, **e**), as in the following example:

The white cat <fr> | *Le chat blanc*

where only the target language tag is appended to the end of the source sentence.

$$\boxed{\mathbf{f}' \langle \text{lang} \rangle \mathbf{e} \langle \text{update} \rangle \mid \mathbf{e}'}$$

The second pattern corresponds to an update task (INS, DEL or SUB) to be used for resynchronizing an initial sentence pair (**f**, **e**) after changing **f** into **f'**, as shown in the following examples:

The white cat <fr> *Le chat* <ins> | *Le chat blanc*
The white cat <fr> *Le chat est blanc* | *Le chat blanc*
The white cat <fr> *Le chat noir* <sub> | *Le chat blanc*

where the edited source sentence $\mathbf{f}' = [\textit{The white cat}]$ is followed by the target language tag <fr>, the initial target sentence **e**, and a tag indicating the edit type that updates the initial source sentence **f**.

$$\boxed{\mathbf{f} \langle \text{lang} \rangle \mathbf{e}_g \mid \mathbf{e}_G}$$

The third pattern corresponds to a bilingual text infilling task (BTI, Xiao et al., 2022). The model is trained to predict the tokens masked in a target sentence \mathbf{e}_g in the context of the source sentence **f**:

The white cat <fr> *Le* <gap> *blanc* | *chat*

where $\mathbf{e}_g = [\textit{Le} \langle \text{gap} \rangle \textit{blanc}]$ is the target sentence with missing tokens to be predicted. The model only generates the masked tokens $\mathbf{e}_G = [\textit{chat}]$.

3.1 Synthetic Data Generation

While large amounts of parallel bilingual data (\mathbf{f}' , \mathbf{e}') exist for many language pairs, the triplets required to train our model are hardly available. We

therefore study ways to generate synthetic triplets of example (\mathbf{f}' , **e**, \mathbf{e}') from parallel data (\mathbf{f}' , \mathbf{e}') for each type of task (INS, DEL, SUB and BTI) introduced above.

Insertion We build examples of initial translations **e** for INS by randomly dropping a segment from the updated target \mathbf{e}' . The length of the removed segment is also randomly sampled with a maximum length of 5 tokens. We also impose that the overall ratio of removed segment does not exceed 0.5 of the length of \mathbf{e}' .

Deletion Simulating deletions requires the initial translation **e** to be an extension of the updated target \mathbf{e}' . To obtain extensions, we employ a NMT model enhanced to fill in gaps (fill-in-gaps). This model is a regular encoder-decoder Transformer model trained with a balanced number of regular parallel examples (TRN) and paraphrase examples (BTI) as detailed in the previous section. We extend training examples (\mathbf{f}' , \mathbf{e}') with a <gap> token inserted in a random position in \mathbf{e}' and use fill-in-gaps to decode these training sentences, as proposed in (Xu et al., 2022). In response, the model predicts tokens that best fill the gap. For instance:

The white cat <fr> *Le chat* <gap> *blanc* \rightsquigarrow *est*

the target extension is therefore $\mathbf{e} = [\textit{Le chat est blanc}]$.

Substitution Similar to deletion, substitution examples are obtained using the same fill-in-gaps model. A random segment is masked from \mathbf{e}' , which is then filled by the model. In inference, the model computes an *n*-best list of substitutions for the mask, and we select the most likely sequence that is not identical to the masked segment. For instance:

The white cat <fr> *Le chat* <gap> \rightsquigarrow [*blanc*; *bleu*; *clair*; *blanche*; ...]

the target substitution is $\mathbf{e} = [\textit{Le chat bleu}]$.

Note that extensions and substitutions generated by fill-in-gaps may be ungrammatical. For instance, the proposed substitution $e = [Le\ chat\ blanche]$ has a gender agreement error. The correct adjective should be "*blanc*" (masculine) instead of "*blanche*" (feminine). This way, the model always learns to produce grammatical e' sentences parallel to f' .

Paraphrase Given sentence pairs (f, e) , we generate e_g by masking a random segment from the initial target sentence e . The length of the masked segment is also randomly sampled with a maximum length of 5 tokens. The target side of these examples (e_G) only contains the masked token(s).

4 Experiments

4.1 Datasets

To train our English-French (En-Fr) models we use the official WMT14 En-Fr corpora⁴ as well as the OpenSubtitles corpus⁵ (Lison and Tiedemann, 2016). A very light preprocessing step is performed to normalize punctuation and to discard examples exceeding a length ratio 1.5 and a limit of [1, 250] measured in words. Statistics of each corpus is reported in Table 1.

<i>Corpora</i>	<i>#Sentences</i>
Europarl v7	2,007,723
Commoncrawl	3,244,152
UN	12,886,831
News Commentary	183,251
Giga French-English	22,520,376
Open Subtitles v18	57,123,540
Total	97,965,873

Table 1: Statistics of training corpora.

For testing, we used the official newstest2014 En-Fr test set made available for the same WMT14 shared task containing 3,003 sentences. All our data is tokenized using OpenNMT tokenizer.⁶ We learn a joint Byte Pair Encoding (Sennrich et al., 2016) over English and French training data with 32k merge operations.

The training corpora used for learning our model consist of well-formed sentences. Most sentences start with a capital letter and end with a punctuation

⁴<https://www.statmt.org/wmt14>

⁵<https://opus.nlpl.eu/OpenSubtitles-v2018.php>

⁶<https://github.com/OpenNMT/Tokenizer>

mark. However, our BiSync editor expects also incomplete sentences, when synchronization occurs before completing the text. To train our model to handle this type of sentences, we lowercase the first character of sentences and remove ending punctuation in both source and target examples with a probability set to 0.05.

4.2 Experimental Settings

Our BiSync model is built using the Transformer architecture (Vaswani et al., 2017) implemented in OpenNMT-tf⁷ (Klein et al., 2017). More precisely, we use the following set-up: embedding size: 1,024; number of layers: 6; number of heads: 16; feedforward layer size: 4,096; and dropout rate: 0.1. We share parameters for input and output embedding layers (Press and Wolf, 2017). We train our models using Noam schedule (Vaswani et al., 2017) with 4,000 warm-up iterations. Training is performed over a single V100 GPU during 500k steps with a batch size of 16,384 tokens per step. We apply label smoothing to the cross-entropy loss with a rate of 0.1. Resulting models are built after averaging the last ten saved checkpoints of the training process. For inference, we use CTranslate2.⁸ It implements custom run-time with many performance optimization techniques to accelerate decoding execution and reduce memory usage of models on CPU and GPU. We also evaluate our model with weight quantization using 8-bit integer (int8) precision, thus reducing model size and accelerating execution compared to the default 32-bit float (float) precision.

5 Evaluation

We evaluate the performance of our synchronization model BiSync compared to a baseline translation model with exactly the same characteristics but trained only on the TRN task over bidirectional parallel data (base). We report performance with BLEU score (Papineni et al., 2002) implemented in SacreBLEU⁹ (Post, 2018) over concatenated En-Fr and Fr-En test sets. For tasks requiring an initial target e , we synthesize e from (f', e') pairs following the same procedures used for generating the training set (see details in Section 2).

Table 2 reports BLEU scores for our two systems on all tasks. The base system is only trained to

⁷<https://github.com/OpenNMT/OpenNMT-tf>

⁸<https://github.com/OpenNMT/CTranslate2>

⁹<https://github.com/mjpost/sacrebleu>. Signature: nrefs:1lcase:mixedlfff:noltok:13alsmooth:expl version:2.0.0

perform regular translations (TRN) for which it obtains a BLEU score of 36.0, outperforming BiSync, which is trained to perform all tasks. This difference can be explained by the fact that BiSync must learn a larger number of tasks than base. When performing INS, DEL and SUB tasks, BiSync vastly outperforms the results of TRN task as it makes good use of the initial translation e . When we use BiSync to generate paraphrases of an initial input (BTI), we obtain a higher BLEU score of 42.6 than the regular translation task (TRN, 34.9). This demonstrates the positive impact of using target side context for paraphrasing.

BLEU	TRN	INS	DEL	SUB	BTI
base	36.0	-	-	-	-
BiSync	34.9	87.9	95.5	78.2	42.6

Table 2: BLEU scores over concatenated En-Fr and Fr-En test sets for all tasks.

Next, we evaluate the ability of our BiSync model to remain close to an initial translation when performing synchronization. Note that for a pleasant editing experience, synchronization should introduce only a minimum number of changes. Otherwise, despite re-establishing synchronization, additional changes may result in losing updates previously performed by the user. To evaluate this capability of our model, we take an initial translation (f , e) and introduce a synthetic update (say f') as detailed in Section 3.1. This update leads to a new synchronization that transforms e into e' . We would like e' to remain as close as possible to e . Table 3 reports TER scores (Snover et al., 2006) between e and e' computed by SacreBLEU.¹⁰ These results indicate that BiSync produces synchronizations significantly closer to initial translations than those produced by base. This also confirms the findings of Xu et al. (2022).

TER ↓	INS	DEL	SUB
base	36.5	43.4	34.9
BiSync	3.3	5.5	5.0

Table 3: TER scores between e and e' issued from different update types. En-Fr and Fr-En test sets are concatenated.

Finally, Table 4 reports inference efficiency for our BiSync model using CTranslate2. We indi-

¹⁰Signature: nrefs:1|case:l|tok:tercom|norm:nolpunct:yes|asian:nolversion:2.0.0

cate differences in model (*Size* and *Speed*) for different quantization, device, batch size and number of threads. Lower memory requirement and higher inference speed can be obtained by using quantization set to int-8 for both GPU and CPU devices, in contrast to float-32. When running on CPUs, additional speedup is obtained with multithreading. Comparable BLEU scores are obtained in all configurations. Note that for the tool presented in this paper, we must retain single batch size and single thread results (bold figures), since synchronization requests are produced for isolated sentences. Therefore, they cannot take advantage of using multiple threads and large batches.

<i>Quant</i>	<i>Dev</i>	<i>BS</i>	<i>Threads</i>	<i>Size</i>	<i>Speed</i>
float	GPU	64	1	232M	10,267
	GPU	1	1		650
	CPU ¹	64	8 × 4		2,007
	CPU ²	1	1		48
int8	GPU	64	1	59M	12,918
	GPU	1	1		738
	CPU ¹	64	8 × 4		2,666
	CPU ²	1	1		118

Table 4: Inference *Speed* and model *Size* when decoding test sets with several settings: quantization (*Quant*), device (*Dev*), batch size (*BS*) and number of *Threads*. Decoding beam size is set to 3. *Speed* is measured in tokens/second. GPU is a single V100 GPU with 32Gb memory. CPU¹ has 32 cores with 86Gb memory and CPU² is an Intel i7-10850H with 32Gb memory.

6 Conclusion and Further Work

In this paper, we presented BiSync, a bilingual writing assistant system that allows users to freely compose text in two languages while always displaying the two monolingual texts synchronized with the goal of authoring two equally good versions of the text. Whenever users make revisions on either text box, BiSync takes into account the initial translation and reduces the number of changes needed in the other text box as much as possible to restore parallelism. BiSync also assists in the writing process by suggesting alternative reformulations for word sequences or alternative translations based on given prefixes. The synchronization process applies several performance optimization techniques to accelerate inference and reduce the memory usage with no accuracy loss, making BiSync usable even on machines with limited computing power.

In the future, we plan to equip BiSync with a

grammatical error prediction model and a better handling of prior revisions: the aim is to enable finer-grained distinction between parts that the system should modify and parts that have already been fixed or that should remain unchanged. Last, we would like to perform user studies to assess the division of labor between users and BiSync in an actual bilingual writing scenario.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable suggestions and comments. This work was performed while the last two authors were affiliated with LISN, CNRS (Orsay). Jitao Xu has been partly funded by SYSTRAN and by a grant (Transwrite) from the Région Ile-de-France. He was granted access to the HPC resources of IDRIS under the allocation 2022-[AD011011580R2] made by GENCI.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Julien Bourdaillet, Stéphane Huet, Philippe Langlais, and Guy Lapalme. 2011. [TransSearch: from a bilingual concordancer to a translation finder](#). *Machine Translation*, 24(3):241.
- Mei-Hua Chen, Shih-Ting Huang, Hung-Ting Hsieh, Ting-Hui Kao, and Jason S. Chang. 2012. [FLOW: A first-language-oriented writing assistant system](#). In *Proceedings of the ACL 2012 System Demonstrations*, pages 157–162, Jeju Island, Korea. Association for Computational Linguistics.
- Alister Cumming. 1989. [Writing expertise and second-language proficiency*](#). *Language Learning*, 39(1):81–135.
- Chung-chi Huang, Ping-che Yang, Keh-jiann Chen, and Jason S. Chang. 2012. [Transahead: A computer-assisted translation and writing tool](#). In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 352–356, Montréal, Canada. Association for Computational Linguistics.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. [OpenNMT: Open-source toolkit for neural machine translation](#). In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.
- Philipp Koehn. 2010. [Enabling monolingual translators: Post-editing vs. options](#). In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 537–545, Los Angeles, California. Association for Computational Linguistics.
- Mariëlle Leijten, Luuk Van Waes, Karen Schriver, and John R. Hayes. 2014. [Writing in the workplace: Constructing documents using multiple digital sources](#). *Journal of Writing Research*, 5(3):285–337.
- Pierre Lison and Jörg Tiedemann. 2016. [OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).
- Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. [Agreement on target-bidirectional neural machine translation](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416, San Diego, California. Association for Computational Linguistics.
- Jihyung Moon, Hyunchang Cho, and Eunjeong L. Park. 2020. [Revisiting round-trip translation for quality estimation](#). In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 91–104, Lisboa, Portugal. European Association for Machine Translation.
- 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.
- 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.
- Ofir Press and Lior Wolf. 2017. [Using the output embedding to improve language models](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. 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 (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

- Matthew Snover, Bonnie Dorr, Rich Schwartz, Linea Micciulla, and John Makhoul. 2006. [A study of translation edit rate with targeted human annotation](#). In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.
- Sriram Venkatapathy and Shachar Mirkin. 2012. [An SMT-driven authoring tool](#). In *Proceedings of COLING 2012: Demonstration Papers*, pages 459–466, Mumbai, India. The COLING 2012 Organizing Committee.
- Mark Wolfersberger. 2003. [L1 to L2 writing process and strategy transfer: A look at lower proficiency writers](#). *TESL-EJ*, 7(2):1–12.
- Yanling Xiao, Lemaoy Liu, Guoping Huang, Qu Cui, Shujian Huang, Shuming Shi, and Jiajun Chen. 2022. [BiTIIMT: A bilingual text-infilling method for interactive machine translation](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1958–1969, Dublin, Ireland. Association for Computational Linguistics.
- Jitao Xu, Josep Crego, and François Yvon. 2022. [Bilingual synchronization: Restoring translational relationships with editing operations](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8016–8030, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Measuring Power and Social Dynamics Between Entities

Maria Antoniak[♣] Anjalie Field[†] Jimin Mun[♡] Melanie Walsh[◇]
 Lauren F. Klein[♣] Maarten Sap^{♡♣}

[♣]Allen Institute for AI [†]Johns Hopkins University [♡]Carnegie Mellon University

[◇]University of Washington [♣]Emory University

<http://github.com/maartensap/riveter-nlp>

Abstract

RIVETER provides a complete easy-to-use pipeline for analyzing verb connotations associated with entities in text corpora. We pre-populate the package with connotation frames of sentiment, power, and agency, which have demonstrated usefulness for capturing social phenomena, such as gender bias, in a broad range of corpora. For decades, lexical frameworks have been foundational tools in computational social science, digital humanities, and natural language processing, facilitating multifaceted analysis of text corpora. But working with verb-centric lexica specifically requires natural language processing skills, reducing their accessibility to other researchers. By organizing the language processing pipeline, providing complete lexicon scores and visualizations for all entities in a corpus, and providing functionality for users to target specific research questions, RIVETER greatly improves the accessibility of verb lexica and can facilitate a broad range of future research.

1 Introduction

Language is a powerful medium that intricately encodes social dynamics between people, such as perspectives, biases, and power differentials (Fiske, 1993). When writing, authors choose how to *portray* or *frame* each person in a text, highlighting certain features (Entman, 1993) to form larger arguments (Fairhurst, 2005). For example, in the screenplay for the 2009 film *Sherlock Holmes*, the authors dramatize a sudden reversal of power by playing on gender stereotypes. First, they describe how “the man with the roses **beckons** Irene forward” (Figure 1), which portrays the character Irene Adler as being lured by the man. After she is trapped, “she **slices** upward with a razor-sharp knife,” reversing the power dynamic. Here, specific word choices shape and then challenge the viewers’ expectations about how the interaction is presumed to unfold.

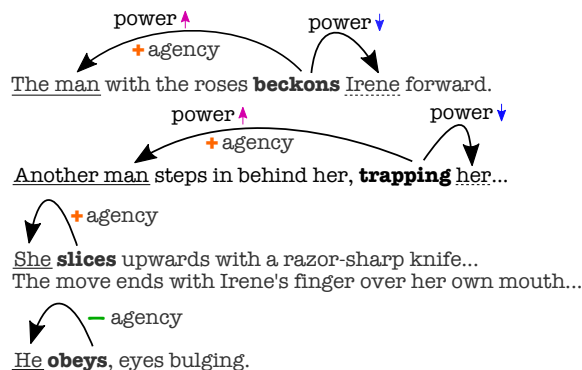


Figure 1: Figure from Sap et al. (2017) illustrating power and agency connotation frames extracted on an excerpt from the *Sherlock Holmes* (2009) film screenplay. Each connotation frame pertains to a **verb predicate** and its agent and theme.

More broadly, an author’s word choices can not only communicate important details about a character or narrative but can also reveal larger social attitudes and biases (Blackstone, 2003; Cikara and Fiske, 2009), shape readers’ opinions and beliefs about social groups (Behm-Morawitz and Mastro, 2008), as well as act as powerful mechanisms to persuade or to induce empathy (Smith and Petty, 1996; Keller et al., 2003). Studying these word choices across large datasets can illuminate domain-specific patterns of interest to scholars in the humanities and social sciences.

Examining verbs—*who* does *what* to *whom*?—is one established approach for measuring the textual portrayal of people and groups of people. Each verb carries connotations that can indicate the social dynamics at play between the subject and object of the verb. Connotation frames (Rashkin et al., 2016; Sap et al., 2017) capture these dynamics by coding verbs with directional scores. For example, these frames might label verbs with who holds power and who lacks power, or who is portrayed with positive or negative sentiment. In the *Sherlock Holmes* scene description, “Another man steps in

behind her, **trapping her**,” the verb *to trap* implies that “the man” has more power over “her” (Figure 1; Sap et al., 2017). These verb lexica have been used successfully to study portrayals in many diverse contexts including films (Sap et al., 2017), online forums (Antoniak et al., 2019), text books (Lucy et al., 2020), Wikipedia (Park et al., 2021), and news articles (Field et al., 2019).

Lexica in general are extremely popular among social science and digital humanities scholars. They are interpretable and intuitive (Grimmer and Stewart, 2013), especially when compared with black-box classification models, and continue to be a go-to resource (e.g., LIWC; Pennebaker et al., 2015). However, verb-based lexica pose specific technical hurdles for those less experienced in software engineering and natural language processing (NLP). These lexica require core NLP skills such as traversing parse trees, identifying named entities and references, and lemmatizing verbs to identify matches. At the same time, the larger research questions motivating their usage require deep domain expertise from the social sciences and humanities.

To meet this need, we introduce **RIVETER**¹, which includes tools to use, evaluate, and visualize verb lexica, enabling researchers to measure power and other social dynamics between entities in text. This package includes a pipeline system for importing a lexicon, parsing a dataset, identifying people or entities, resolving coreferences, and measuring patterns across those entities. It also includes evaluation and visualization methods to promote grounded analyses within a targeted dataset. We release RIVETER as a Python package, along with Jupyter notebook demonstrations and extensive documentation aimed at social science and humanities researchers.

To showcase the usefulness of this package, we describe two case studies: (1) power differentials and biases in GPT-3 generated stories and (2) gender-based patterns in the novel *Pride and Prejudice*. The first study provides a proof-of-concept; dyads with predetermined power differentials are used to generate stories, and we are able to detect these distribution shifts using RIVETER. The second study zooms in on a particular author, text,

¹The name Riveter is inspired by “Rosie the Riveter,” the allegorical figure who came to represent American women working in factories and at other industrial jobs during World War II. Rosie the Riveter has become an iconic symbol of power and shifting gender roles—subjects that the Riveter package aims to help users measure and explore by combining (or *riveting*) components into a pipeline.

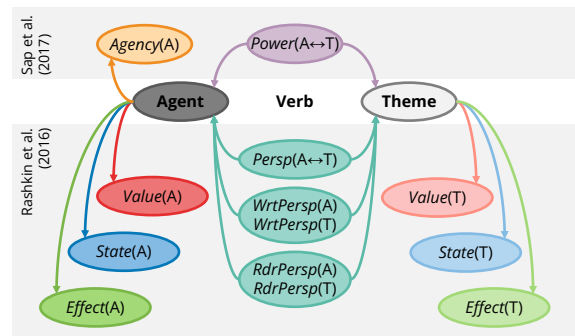


Figure 2: A verb predicate can connote various sentiment, power, and agency levels for its agent and theme; connotation frames distill these into six relation types (§2.3; each type is colored in a different hue).

and social setting, examining how a 19th century novelist both portrayed and subverted gender roles. These case studies highlight the diverse contexts and research questions for which this package can be used across human and machine-generated text, and across the social sciences and the humanities.

2 Background: Verb Lexica & Connotation Frames

2.1 Verb Predicates & Frame Semantics

Understanding the events and states described in sentences, i.e., *who* does *what* to *whom*, has been a central question of linguistics since first conceptualized by Indian grammarian Pāṇini between the 4th and 6th century BCE. Today, verb predicates and their relation to other words in a sentence are still a key focus of linguistic analyses, e.g., dependency parsing (Tesnière, 2015; Nivre, 2010) and semantic role labeling (Gildea and Jurafsky, 2000).

To model how one understands and interprets the events in a sentence, Fillmore (1976) introduced *frame semantics*, arguing that understanding a sentence involves knowledge that is evoked by the concepts in the sentence. This theory inspired the task of *semantic role labeling* (Gildea and Jurafsky, 2000), which categorizes how words in a sentence relate to the main verb predicate via frame semantics. This task defines *thematic roles* with respect to an EVENT (i.e., the verb predicate): the AGENT that causes the EVENT (loosely, the subject of the verb), and the THEME that is most directly affected by the EVENT (loosely, the object of the verb).

Work	Usage
Rashkin et al. (2016)	Analyzing political leaning and bias in news articles.
Sap et al. (2017)	Analyzing gender bias in portrayal of characters in movie scripts.
Rashkin et al. (2017)	Analyzing public sentiment (and multilingual extension of Rashkin et al. (2016))
Volkova and Jang (2018)	Improving the detection of fake news & propaganda.
Ding and Riloff (2018)	Detecting affective events in personal stories.
Field et al. (2019)	Analyzing power dynamics of news portrayals in #MeToo stories.
Antoniak et al. (2019)	Analyzing the power dynamics in birthing stories online.
Lucy et al. (2020)	Analyzing the portrayal of minority groups in textbooks.
Mendelsohn et al. (2020)	Analyzing the portrayal of LGBTQ people in the New York Times.
Ma et al. (2020)	Text rewriting for mitigating agency gender bias in movies.
Park et al. (2021)	Comparing affect in multilingual Wikipedia pages about LGBT people
Lucy and Bamman (2021)	Analyzing gender biases in GPT3-generated stories.
Gong et al. (2022)	Quantifying gender biases and power differentials in Japanese light novels
Saxena et al. (2022)	Examining latent power structures in child welfare case notes
Borchers et al. (2022)	Measuring biases in job advertisements and mitigating them with GPT-3
Stahl et al. (2022)	Joint power-and-agency rewriting to debias sentences.
Wiegand et al. (2022)	Identifying implied prejudice and social biases about minority groups
Giorgi et al. (2023)	Examining the portrayal of narrators in moral and social dilemmas

Table 1: Examples of usages of connotation frames in NLP and CSS literature.

2.2 Connotation Frames of Sentiment, Power, and Agency

While frame semantics was originally meant to capture broad meaning that arises from interpreting words in the context of what is known (Fillmore, 1976), many linguistic theories have focused solely on denotational meaning (Baker et al., 1998; Palmer et al., 2005), i.e., examining only what is present in the sentence. In contrast, the *implied or connoted meaning* has received less attention, despite being crucial to interpreting sentences.

Connotation frames, introduced by Rashkin et al. (2016), were the first to model the connotations of verb predicates with respect to an AGENT and THEME’s value, sentiment, and effects (henceforth, sentiment connotation frames). Shortly thereafter, Sap et al. (2017) introduced the *power and agency connotation frames* (Figure 2), which model the power differential between the AGENT and the THEME, as well as the general agency that is attributed to the AGENT of the verb predicate.²

For both sets of connotation frames, the authors released a lexicon of verbs with their scores. Verbs were selected based on their high usage in corpora of choice: frequently occurring verbs from a corpus of New York Times articles (Sandhaus, 2008) for sentiment connotation frames, and frequently occurring verbs in a movie script corpus (Gorinski and Lapata, 2015) for the power and agency frames. Each verb was annotated for each dimen-

sion by crowdworkers with AGENT and THEME placeholders (“*X implored Y*”).

Since their release, connotation frames have been of increasing interest to researchers working in disciplines like cultural analytics and digital humanities communities. They have given these researchers a flexible and interpretable way to examine the framing of interpersonal dynamics across a wide range of datasets and research questions (Table 1). Additionally, the frames have been incorporated into the 2023 edition of the textbook *Speech and Language Processing* (Jurafsky and Martin, 2023).

2.3 Connotation Frame Dimensions

Given a predicate verb v describing an EVENT and its AGENT a and THEME t , connotation frames capture several implied relations along sentiment, power, and agency dimensions. Each of these relations has either a positive \oplus , neutral \ominus , or negative \ominus polarity. We describe here a set of six example relations included in RIVETER; for a fuller discussion of each of these relations and their definitions, see Rashkin et al. (2016) and Sap et al. (2017).

Effect denotes whether the event described by v has a positive or negative effect on the agent a or the theme t . For example, in Figure 1, another man “trapping” Irene has a negative effect on her ($Effect(t) = \ominus$).

Value indicates whether the agent or theme are presupposed to be of value by the predicate v . For example, when someone “guards” an object,

²While the value, sentiment, effects, and power relations require a verb to be transitive, the agency dimension is present with intransitive verbs as well.

this presupposes that the object has high value ($Value(t) = \oplus$).

State captures whether the likely mental state of the AGENT or THEME as a result of the EVENT. For example, someone “suffering” likely indicates a negative mental state ($State(a) = \ominus$).

Perspective is a set of relations that describe the sentiment of the AGENT towards the THEME and vice versa ($Persp(a \leftrightarrow t)$). It also describes how the writer perceives the AGENT and THEME ($WrtPersp(a)$, $WrtPersp(t)$), as well as how the reader likely feels towards them ($RdrPersp(a)$, $RdrPersp(t)$).

Power distills the power differential between the AGENT and THEME of the EVENT (denoted as $Power(a \leftrightarrow t)$ for shorthand). For example, when a man “traps” Irene, he has power over Irene ($Power(a) = \oplus$ and $Power(t) = \ominus$). In the implementation of RIVETER, we convert the positive \oplus , neutral \ominus , or negative \ominus polarities into categorical scores ($\{-1, 0, +1\}$), as described in §3.2, to facilitate aggregation over entities.

Agency denotes whether the AGENT of the EVENT has agency, i.e., is decisive and can act upon their environment. For example, Irene “slicing” connotes high agency ($Agency(a) = \oplus$), whereas the man “obeying” connotes low agency ($Agency(a) = \ominus$). We convert these categories to numbers as described for *Power* above.

3 RIVETER Design & Implementation

3.1 Challenges Addressed

Unlike lexica that require only string matching, verb lexica indicating relations between the AGENT and THEME also require parsing, lemmatization, named entity recognition, and coreference resolution. These are standard pieces of NLP pipelines, but each piece requires background knowledge in linguistics, NLP, and algorithms that inform library choices and merging of outputs; this can pose a challenge for researchers without extensive NLP knowledge or training. RIVETER substantially lowers the implementation burden and text-processing knowledge required for using verb lexica by addressing the following three challenges.

Familiarity with Using NLP Tools The increasing availability of NLP packages has resulted in numerous existing packages for core NLP pipelines.

We reviewed the performance (considering accuracy, speed, and ease of installation) of available tools and pre-selected optimal text processing pipelines for RIVETER, eliminating the need for users to be familiar with and decide between available text processing tools. We also provide documentation on incorporated packages and extensive demonstrations.

Interfacing between NLP Tools Even if one is familiar with individual tools, like parsers or entity recognizers, connecting outputs from one tool to another tool requires an additional engineering skill set. Traversing a parse tree to find semantic triples and then matching these triples to clusters from a coreference resolution engine is not a straightforward process for a researcher with less expertise in programming and software engineering. To address this challenge, we (a) provide a system that connects these pipeline pieces for the user while also (b) providing functionality to explore the outputs of each individual system.

Interpreting Results Lexical methods can offer flexibility and interpretability not found in other NLP methods, but even so, validating and exploring lexical results can be challenging. Proper validation is not consistent even in NLP research using lexicon-based methods (Antoniak and Mimno, 2021). To address this challenge, we provide methods to explore the results numerically and visually, enabling users to quickly produce plots, calculate aggregate scores, identify contributing verbs and documents, and measure their results’ stability.

3.2 System Description

Illustrated in Figure 3, RIVETER takes in a set of documents as input, and returns a set of scores for each entity appearing as an AGENT or THEME in the target dataset. Under the hood, RIVETER parses the documents, resolves coreference clusters, finds entity mentions, extracts AGENT-EVENT-THEME triples, and computes lexicon scores. We verify our implementation through hand-constructed unit tests and testing of large and small corpora. We describe each of these components below.

Named Entity Recognition and Coreference Resolution Our package first parses a given document to find clusters of mentions that relate to entities. We extract general coreference clusters, which we cross-reference with mentions of entities labeled by a named entity recognition (NER) sys-

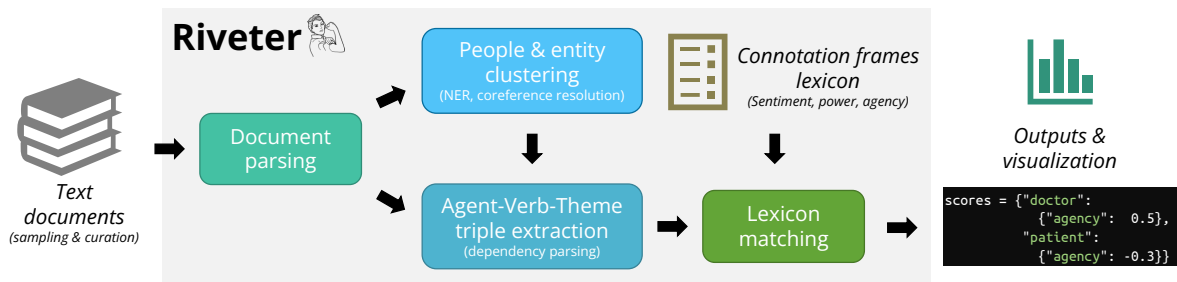


Figure 3: A visualization of the RIVETER pipeline components and their connections.

tem, as well as a list of general people referents (containing pronouns, professions, etc). Coreference cluster mentions that overlap with a mention of an entity are then passed to the verb and relation identification module.

In our implementation, we currently use spaCy for NER, and the spaCy add-on *neuralcoref*³ library for coreference resolution. These libraries are well-supported, have fast run-times relative to similar systems, and do not require GPU access, which are not accessible to many researchers, especially outside of computing disciplines.

Lexicon Loading We include two lexica by default: connotation frames from Rashkin et al. (2016) and frames of power and agency from Sap et al. (2017). These lexica come included in the package, and the user can select between the lexica and their dimensions (see §2.3 for dimension descriptions), as well as use their own custom lexica.

For the verb lexicon from Rashkin et al. (2016) and for custom lexicons, each verb has a numerical score (ranging from -1 to 1) for each of AGENT and THEME. If a lexicon only has scores for AGENT or THEME, the other scores are set to 0. For the verb lexicon from Sap et al. (2017), we convert the categorical labels to numerical scores, so that each verb has a score of $+1$, 0 , or -1 for each of the AGENT and THEME.

For example, in the verb lexicon from Sap et al. (2017), the verb “amuse” is labeled as having positive agency ($Agency(a) = \oplus$) and power for the THEME ($Power(a) = \ominus, Power(t) = \oplus$). In this lexicon, agency was coded only for the AGENT, so we convert the categorical label to a score of $+1$ for the AGENT and 0 for the THEME. For power, we convert the categorical label to -1 for the AGENT and $+1$ for the THEME.

RIVETER also allows the use of custom lexica. We include a loading function for any verb lexicon

formatted in the style of Rashkin et al. (2016). This requires a file listing verbs and their agent and theme scores, which should be positive and negative numbers. This functionality is especially important when using previous lexica on new datasets, as this allows users to customize those lexica for new contexts, simply by updating or adding to the included lexicon files.

Verb Identification and Entity Relation We extract the lemmas of all verbs and match these to the lexicon verbs. After identifying semantic triples (the AGENT or subject (*nsubj*) and THEME or direct object (*dobj*) of each verb) using the spaCy dependency parser, we search for matches to the NER spans identified in §3.2. We track the frequencies of these for the canonical entity, using the converted scores.

Exploration and Visualization We provide functionality for users to easily view lexicon scores for entities in their input text. To maximize utility, we focus on facilitating analyses established in prior work (e.g., Table 1). This functionality includes:

- retrieving the overall verb lexicon scores for all entities identified in the entire input corpora (`get_score_totals`),
- retrieving the overall verb lexicon scores for all entities identified in a specific document (`get_scores_for_doc`),
- generating bar plots of scores for entities in the dataset (e.g., filtering for the top-scored entities) (`plot_scores`) or in a specific document (`plot_scores_for_doc`).

We additionally provide functionality to reduce the opacity of lexicon scores and allow users to examine specific findings in more depth or conduct error checking. These functions include:

³<https://github.com/huggingface/neuralcoref>

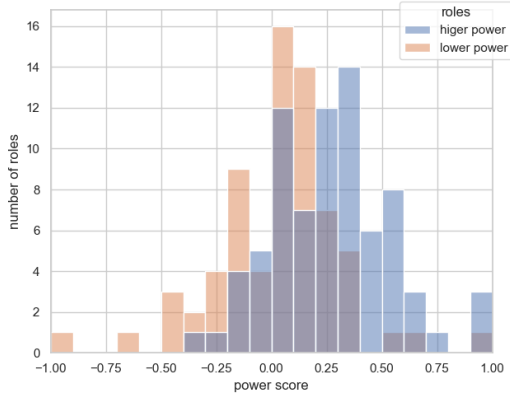


Figure 4: Number of roles with corresponding power scores color coded by assignment of higher and lower power roles over 85 short stories sampled from GPT-3.5.

- generating heat map plots for the verbs most frequently used with a user-specified entity (`plot_verbs_for_persona`),
- retrieving all mentions associated with a specified entity, after co-reference resolution (`get_persona_cluster`),
- retrieving various additional counts, including number of lexicon verbs in a document, all entity-verb pairs in a document, number of identified entities, etc.
- retrieving all documents that matched an entity or verb (`get_documents_for_verb`),
- bootstrapping the dataset to examine stability across samples (`plot_verbs_for_persona`).

4 Case Studies Across Cultural Settings

4.1 Machine Stories with Power Dyads

As our first case study, we examine lexicon-based power differentials in machine-generated stories about two characters with a predetermined power asymmetry, as in [“doctor”, “patient”], [“teacher”, “student”], and [“boss”, “employee”]. By generating stories about entities with predetermined power asymmetries, this serves as a proof-of-concept for RIVETER; we expect to measure power scores in the predetermined directions.

Given a set of 32 pairs of roles, we obtain 85 short stories from GPT-3.5 (Ouyang et al., 2022) (see Appendix A for details). We then scored each of the characters with assigned roles using RIVETER and aggregated the scores for higher power roles and lower power roles using their names given by GPT-3.5.

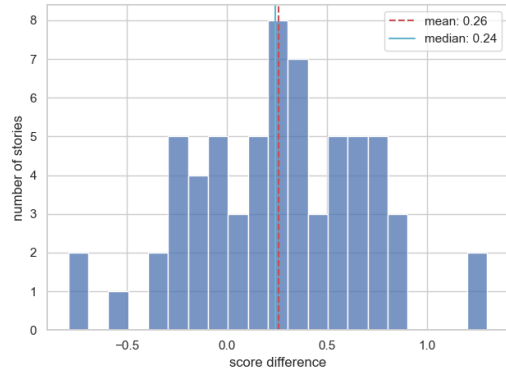


Figure 5: Number of stories with corresponding power score differences calculated by taking $Power(e^+) - Power(e^-)$ in each generated story where e^+ and e^- are pairs of entities predetermined to have high and low power roles (e.g., [“doctor”-“patient”]).

Results As seen in Figure 4, higher power roles have a distribution shifted toward greater power scores than lower power roles. The mean score of the higher power roles was 0.265 and that of lower power roles was 0.0364. A t-test also shows statistical significance ($p < 0.05$). From these results we can conclude that the stories generated by GPT-3.5 reflect the power dynamics in the relations given in the prompt, and our framework captures this expected phenomena.

The differences in power scores show similar results in Figure 5. These differences were calculated only for the stories where both higher and lower power figures had been scored. The mean and median were both positive, 0.26 and 0.24.

Analyzing the stories with both positive and negative score differences (see Table 3 in the Appendix) further confirms the results of our framework. For example, the third story of the table shows negative score difference between higher powered agent and lower powered agent. However, looking at it more closely we can see that despite the assigned role as an interviewee, Emily shows greater power. In the sentence “Paul thanked Emily for her time and wished her luck,” both *thank* and *wish* from our lexicon give Emily greater power than Paul. Thus we can analyze the power dynamics between characters more accurately, taking into account more context than just assigned roles.

4.2 Gender Differences in *Pride and Prejudice*

Jane Austen’s 1813 novel *Pride and Prejudice* is famous for its depiction of gender and class relations in 19th century England. Using the entity

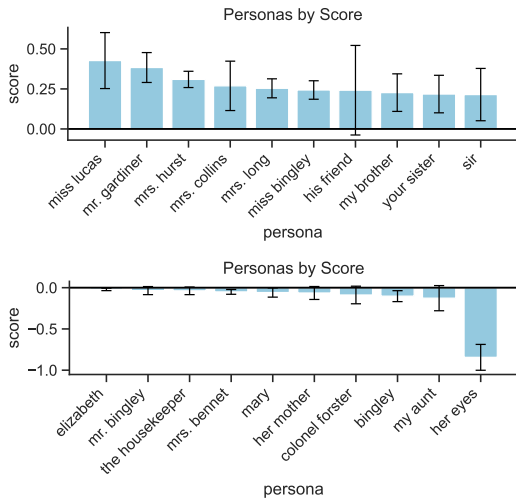


Figure 6: Means and standard deviations of power scores across 20 bootstrapped samples of *Pride and Prejudice*, using RIVETER’s sampling and visualization functions. Results are shown for three custom pronoun groups captured using RIVETER’s entity discovery and coreference resolution capabilities.

recognition and coreference resolution capabilities of RIVETER, we can identify which characters are framed as having or lacking power, and by examining the power relations between classes of third person pronouns, we can trace how gender hierarchies are enacted and subverted by Austen, through the actions of her characters.

Results Figure 6 shows the entities identified by RIVETER that have the highest and lowest power scores, using the lexicon from Sap et al. (2017). Characters like Miss Lucas have higher power scores, while characters like Elizabeth have lower power scores. By using RIVETER’s functionality to pull out the documents contributing to an entity’s score, we find that a mistaken entity, “her eyes,” indeed often occurs in low power roles, as in “Miss Bingley fixed her eyes on face,” providing an intuitive validation of the results. This plot also shows the standard deviation across bootstrapped samples of the novel, indicating the overlapping instability of many of the power scores for this single novel.

Figure 7 shows the lexicon verbs contributing most frequently to each pronoun group’s power score. For example, we see that feminine pronouns are frequently used as subjects of the verb “hear”—emphasizing women’s low-power role of waiting to hear news. We also observe that while feminine pronouns are often placed in high-power positions at rates similar to masculine pronouns, they have

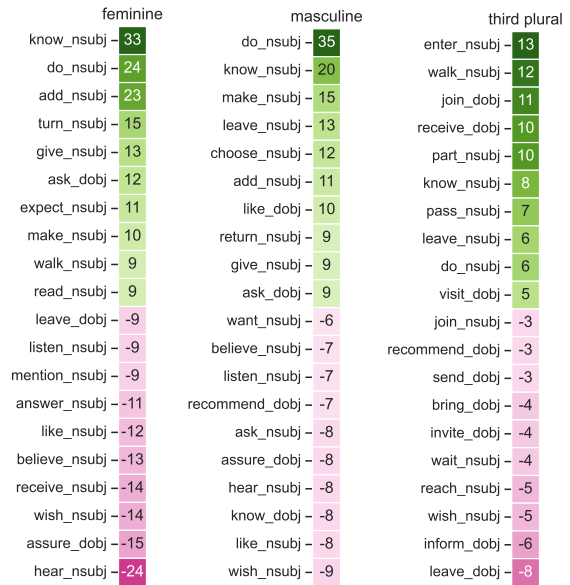


Figure 7: Verb counts for pronoun groups in *Pride and Prejudice*, using RIVETER’s visualization functions. Green cells indicate verbs where the pronoun group **has** power, while pink cells indicate verbs where the pronoun group **lacks** power. Labels include both the verb and the position (*nsubj* or *dobj*) of the pronoun group, and the pronoun groups were captured via RIVETER’s customizable entity discovery function.

higher frequencies for low-power positions. In other words, in Austen’s world, masculine and feminine entities both engage in high-power actions, but feminine entities engage in more low-power actions. Arguably, though, some of the low-power positions are used by the feminine entities to obtain power, e.g., by “hearing” news or eavesdropping on others, the feminine entities can learn information that informs their future decisions and strategies.

5 Ethics and Broader Impact

RIVETER comes with some risks and limitations. This package is targeted only at English-language texts; we have not included non-English lexica in the tool nor do we expect the parsing, named entity recognition, and coreference resolution to directly translate to other languages. While related lexica exist for non-English languages (e.g., Klenner et al. (2017) (German), Rashkin et al. (2017) (extension to 10 European languages)), the generalizability of RIVETER is limited to English-language settings.

The results of RIVETER are only as reliable as the corpora and lexica used as input (and their relationships to one another). We have emphasized interpretability in designing this package, encourag-

ing users to examine their results at different levels of granularity. However, there are still dangers of biases “baked-in” to the data, via its sampling and curation, or to the lexica, in the choice of terms and their annotations by human workers. Lexica that are useful in one setting are not always useful in other settings, and we encourage researchers to validate their lexica on their target corpora.

Drawing from a framework describing the roles for computational tools in social change (Abebe et al., 2020), we believe that RIVETER can fill multiple important roles. First, it can act as a *diagnostic*, measuring social biases and hierarchies across large corpora, as in Mendelsohn et al. (2020) where dehumanization of LGBTQ+ people was measured across news datasets and time. RIVETER can also act as a *formalizer*, allowing researchers to examine the specific words used by authors, adding concrete and fine-grained evidence to the constructions of broader patterns, as in Antoniak et al. (2019) where the supporting words were used to characterize healthcare roles during childbirth. Finally, RIVETER can act as a *synecdoche* by bringing new attention and perspectives to long-recognized problems, as in Sap et al. (2017) where renewed attention was given to gender biases in film.

6 Acknowledgements

We thank Hannah Rashkin for the invaluable feedback on the paper and for creating the original connotation frames. We also thank our anonymous reviewers whose comments helped us improve both this paper and RIVETER.

References

- Rediet Abebe, Solon Barocas, Jon Kleinberg, Karen Levy, Manish Raghavan, and David G. Robinson. 2020. *Roles for computing in social change*. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20*, page 252–260, New York, NY, USA. Association for Computing Machinery.
- Maria Antoniak and David Mimno. 2021. *Bad seeds: Evaluating lexical methods for bias measurement*. 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 1889–1904, Online. Association for Computational Linguistics.
- Maria Antoniak, David Mimno, and Karen Levy. 2019. *Narrative paths and negotiation of power in birth stories*. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW).
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. *The Berkeley FrameNet project*. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.
- Elizabeth Behm-Morawitz and Dana E Mastro. 2008. Mean girls? The influence of gender portrayals in teen movies on emerging adults’ gender-based attitudes and beliefs. *Journalism & Mass Communication Quarterly*, 85(1):131–146.
- Amy M Blackstone. 2003. *Gender Roles and Society*.
- Conrad Borchers, Dalia Gala, Benjamin Gilbert, Eduard Oravkin, Wilfried Bounsi, Yuki M Asano, and Hannah Kirk. 2022. *Looking for a handsome carpenter! Debiasing GPT-3 job advertisements*. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 212–224, Seattle, Washington. Association for Computational Linguistics.
- Mina Cikara and Susan T Fiske. 2009. *Warmth, competence, and ambivalent sexism: Vertical assault and collateral damage*, volume 21. American Psychological Association, xvii, Washington, DC, US.
- Haibo Ding and Ellen Riloff. 2018. Weakly supervised induction of affective events by optimizing semantic consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of Communication*, 43(4):51–58.
- Gail T. Fairhurst. 2005. Reframing the art of framing: Problems and prospects for leadership. *Leadership*, 1:165 – 185.
- Anjalie Field, Gayatri Bhat, and Yulia Tsvetkov. 2019. *Contextual affective analysis: A case study of people portrayals in online metoo stories*. *Proceedings of the International AAAI Conference on Web and Social Media*, 13(01):158–169.
- Charles J. Fillmore. 1976. *Frame semantics and the nature of language**. *Annals of the New York Academy of Sciences*, 280(1):20–32.
- Susan T Fiske. 1993. Controlling other people: The impact of power on stereotyping. *American Psychologist*, 48(6):621–628.
- Daniel Gildea and Daniel Jurafsky. 2000. *Automatic labeling of semantic roles*. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 512–520, Hong Kong. Association for Computational Linguistics.
- Salvatore Giorgi, Ke Zhao, Alexander H Feng, and Lara J Martin. 2023. Author as character and narrator: Deconstructing personal narratives from the r/amttheasshole reddit community. In *ICWSM 2023*.

- Xiaoyun Gong, Yuxi Lin, Ye Ding, and Lauren Klein. 2022. Gender and power in Japanese light novels. *Proceedings of Computational Humanities Research Conference*.
- Philip John Gorinski and Mirella Lapata. 2015. [Movie script summarization as graph-based scene extraction](#). In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.
- Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297.
- Daniel Jurafsky and James Martin. 2023. *Speech and Language Processing, 3rd Edition*. Prentice Hall.
- Punam Anand Keller, Isaac M Lipkus, and Barbara K Rimer. 2003. Affect, framing, and persuasion. *J. Mark. Res.*, 40(1):54–64.
- Manfred Klenner, Don Tuggener, and Simon Clematide. 2017. [Stance detection in Facebook posts of a German right-wing party](#). In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 31–40, Valencia, Spain. Association for Computational Linguistics.
- Li Lucy and David Bamman. 2021. [Gender and representation bias in GPT-3 generated stories](#). In *Proceedings of the Third Workshop on Narrative Understanding*, pages 48–55, Virtual. Association for Computational Linguistics.
- Li Lucy, Dorottya Demszky, Patricia Bromley, and Dan Jurafsky. 2020. Content analysis of textbooks via natural language processing: Findings on gender, race, and ethnicity in texas us history textbooks. *AERA Open*, 6(3):2332858420940312.
- Xinyao Ma, Maarten Sap, Hannah Rashkin, and Yejin Choi. 2020. [PowerTransformer: Unsupervised controllable revision for biased language correction](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7426–7441, Online. Association for Computational Linguistics.
- Julia Mendelsohn, Yulia Tsvetkov, and Dan Jurafsky. 2020. A framework for the computational linguistic analysis of dehumanization. *Frontiers in Artificial Intelligence*, 3:55.
- Joakim Nivre. 2010. Dependency parsing. *Language and Linguistics Compass*, 4(3):138–152.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. [The Proposition Bank: An annotated corpus of semantic roles](#). *Computational Linguistics*, 31(1):71–106.
- Chan Young Park, Xinru Yan, Anjalie Field, and Yulia Tsvetkov. 2021. [Multilingual contextual affective analysis of lgbt people portrayals in wikipedia](#). *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):479–490.
- James W. Pennebaker, Roger J. Booth, Ryan L. Boyd, and Martha E. Francis. 2015. Linguistic inquiry and word count: LIWC 2015.
- Hannah Rashkin, Eric Bell, Yejin Choi, and Svitlana Volkova. 2017. [Multilingual connotation frames: A case study on social media for targeted sentiment analysis and forecast](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 459–464, Vancouver, Canada. Association for Computational Linguistics.
- Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. [Connotation frames: A data-driven investigation](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 311–321, Berlin, Germany. Association for Computational Linguistics.
- Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.
- Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. [Connotation frames of power and agency in modern films](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2329–2334, Copenhagen, Denmark. Association for Computational Linguistics.
- Devansh Saxena, Seh Young Moon, Dahlia Shehata, and Shion Guha. 2022. Unpacking invisible work practices, constraints, and latent power relationships in child welfare through casenote analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–22.
- Stephen M Smith and Richard E Petty. 1996. Message framing and persuasion: A message processing analysis. *Pers. Soc. Psychol. Bull.*, 22(3):257–268.
- Maja Stahl, Maximilian Spliethöver, and Henning Wachsmuth. 2022. [To prefer or to choose? generating agency and power counterfactuals jointly for gender bias mitigation](#). In *Proceedings of the Fifth Workshop on Natural Language Processing and Computational Social Science (NLP+CSS)*, pages 39–51,

Abu Dhabi, UAE. Association for Computational Linguistics.

Lucien Tesnière. 2015. *Elements of structural syntax*. John Benjamins Publishing Company.

Svitlana Volkova and Jin Yea Jang. 2018. [Misleading or falsification: Inferring deceptive strategies and types in online news and social media](#). In *Companion Proceedings of the The Web Conference 2018, WWW '18*, page 575–583, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Michael Wiegand, Elisabeth Eder, and Josef Ruppenhofer. 2022. [Identifying implicitly abusive remarks about identity groups using a linguistically informed approach](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5600–5612, Seattle, United States. Association for Computational Linguistics.

A Appendix: GPT-3.5 Generation Setup

Here we further detail our steps to evaluate our framework as discussed in Section 4.1. To generate characters with clear roles, names, and power differences, we used 32 dyadic relation pairs with explicit power asymmetry in our prompt. The full list of relations are shown in Table 2. The following is an example of the prompt used to generate such stories:

Tell me a short story about a doctor and a patient, and give them names.

doctor's name:

Using text-davinci-003 model, we generated 3 stories per pair with temperature set to 0.7 and max tokens set to 256. After cleaning ill-formatted results, we analyzed a total of 85 stories. A few examples of the generated stories along with the power scores of the characters are shown in Table 3.

(doctor, patient)	(teacher, student)
(interviewer, interviewee)	(parent, child)
(employer, employee)	(boss, subordinate)
(manager, worker)	(landlord, tenant)
(judge, defendant)	(supervisor, intern)
(therapist, client)	(owner, customer)
(mentor, mentee)	(politician, voter)
(rich, poor)	(elder, younger)
(artist, critic)	(host, guest)
(preacher, parishioner)	(expert, novice)
(counselor, advisee)	(coach, athlete)
(lender, borrower)	(king, subject)
(seller, buyer)	(umpire, player)
(printer, writer)	(sponsor, sponsored)
(producer, actor)	(legislator, citizen)
(commander, soldier)	(executor, beneficiary)

Table 2: Full list of dyadic roles

GPT3.5 Generated Story		
Dyadic Roles	Power Scores	Power Score Difference
<p>Alan was a successful CEO of a large company. He had a sharp eye for detail and was known for his strict yet fair leadership style. One day, Alan noticed that one of his subordinates, Zara, was not performing up to the standards he expected. He called her into his office and asked her why her work wasn't up to par. Zara admitted that she was feeling overwhelmed with her workload and was having trouble keeping up. She asked Alan if he could help her prioritize her tasks and provide her with additional resources to help her complete her work. Alan was impressed by Zara's initiative and agreed to help her. He rearranged her tasks and assigned her additional help from other staff members. He also gave her some helpful tips on how to manage her workload more effectively. Zara was grateful for Alan's help and was able to complete her work on time and to a high standard. Alan was pleased with her performance and decided to reward her with a promotion. From then on, Zara and Alan had an excellent working relationship. Zara's hard work and resilience was an example to her colleagues, and Alan's willingness to help her out was a</p> <p>(boss, subordinate)</p>	Alan: 0.4, Zara: -0.07	0.47
<p>Mr. Jones was a landlord who owned an apartment building in a small town. He had been a landlord for many years, and he was known for his strict rules and no-nonsense attitude. Ms. Lopez was a single mother who had recently moved into the building. She was looking for a safe place to raise her young daughter, and Mr. Jones' building seemed to fit the bill. She was a bit intimidated by his strict rules, but she was willing to abide by them if it meant living in a safe and secure place. At first, things went smoothly between Mr. Jones and Ms. Lopez. He made sure the building was well maintained, and she was a model tenant, paying her rent on time every month. However, one day Ms. Lopez noticed that her hot water heater had stopped working. She immediately contacted Mr. Jones to let him know, but he refused to repair it. She was frustrated and angry, but she didn't want to risk getting evicted, so she kept quiet. After a few weeks, the hot water heater finally broke down completely and Ms. Lopez had no choice but to contact Mr. Jones again.</p> <p>(landlord, tenant)</p>	Mr. Jones: 0.42, Ms. Lopez: 0.09	0.33
<p>Paul was an experienced interviewer, but Emily was his most challenging interviewee yet. He had seen her resume online and was impressed with her qualifications. He had high hopes for her. When Emily arrived for the interview, Paul immediately noticed her confidence. She was wearing a suit and had a bright smile on her face. She was clearly prepared and ready for the interview. Paul asked Emily several questions about her past experience and qualifications. She answered each one with poise and detail. She was articulate and could back up her answers with examples. Paul was very impressed with Emily's performance. He could tell she was a strong candidate and he was confident she would do well in the job. At the end of the interview, Paul thanked Emily for her time and wished her luck. He was sure she would be a great addition to the company. Emily left the interview feeling pleased with how it went. She was confident she had done her best and was sure she had impressed Paul. The next day, Paul gave Emily an offer for the job. She happily accepted and started a few weeks later. Paul and Emily had a successful working</p> <p>(interviewer, interviewee)</p>	Paul: 0.07, Emily: 0.25	-0.17

Table 3: Examples of GPT3.5 generated stories and the power scores of the characters

Fast Whitespace Correction with Encoder-Only Transformers

Hannah Bast¹ and Matthias Hertel^{1,2} and Sebastian Walter^{1*}

¹University of Freiburg

Department of Computer Science

Freiburg, Germany

{bast,swalter}@cs.uni-freiburg.de

²Karlsruhe Institute of Technology

Institute for Automation and Applied Informatics

Karlsruhe, Germany

matthias.hertel@kit.edu

Abstract

The goal of whitespace correction is to fix space errors in arbitrary given text. For example, given the text *whi te space correc tio nwithTransf or mers*, produce *whitespace correction with Transformers*. We compare two Transformer-based models, a character-level encoder-decoder model and a byte-level encoder-only model. We find that the encoder-only model is both faster and achieves higher quality. We provide an easy-to-use tool that is over 900 times faster than the previous best tool, with the same high quality. Our tool repairs text at a rate of over 200 kB/s on GPU, with a sequence-averaged F₁-score ranging from 87.5% for hard-to-correct text up to 99% for text without any spaces.

1 Introduction

Most natural language processing applications assume a segmentation of the text into words. In English (and many other languages), this segmentation is typically achieved by splitting the text at space characters (and a few additional rules). However, many texts contain a significant amount of space errors, that is, spurious spaces or missing spaces. These can be due to OCR errors, imperfect extraction from PDF files, or typing errors. See Bast et al. (2021) for a more thorough discussion of the sources of such errors.

We consider the following *whitespace correction* problem: given a text in natural language, with an arbitrary amount of missing or spurious spaces, compute a variant of the text with correct spacing. The text may contain spelling errors, which make the task more difficult, but it's not part of the problem to correct them. However, as shown in Bast et al. (2021), with spaces repaired, spelling-correction algorithms do a much better job.

The best previous methods for whitespace correction achieve high F₁-scores, however, at the

price of very slow processing speeds; see Section 2. This is a major obstacle for the practical use of such systems for large amounts of text. Our goal in this work is to provide a practical tool with a much higher speed, without sacrificing the high quality.

1.1 Contributions

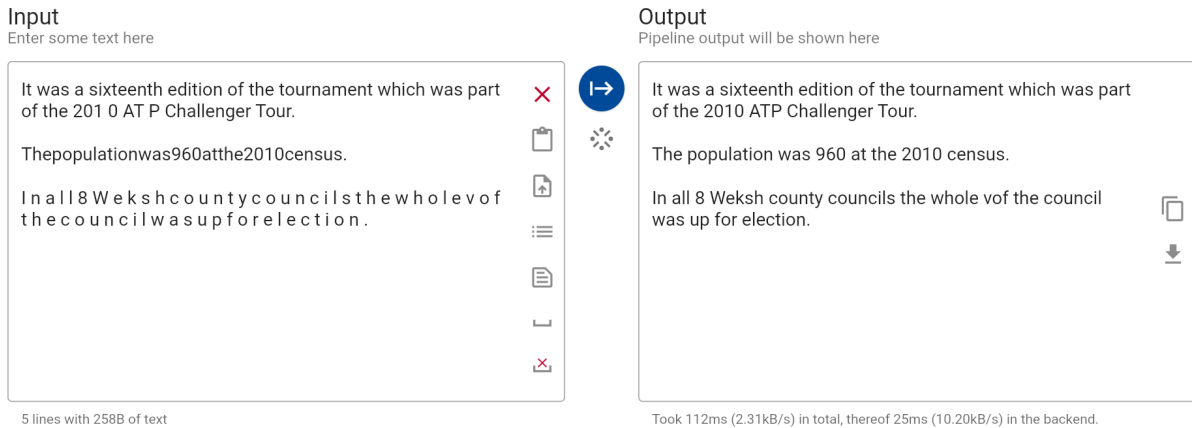
We consider these as our main contributions:

- We provide a practical method for whitespace correction that is over 900 times faster than the best previous method, with the same high quality across a wide selection of benchmarks. On an A100 GPU, our tool repairs text at over 200 kB/s with a sequence-averaged F₁-score ranging from 87.5% (for hard-to-correct text) to 99% (for text without any spaces).
- We compare two Transformer-based models: a character-level encoder-decoder model (that outputs the repaired sequence), and a byte-level encoder-only model (that predicts for each gap between two characters whether there should be a space or not). Both models take existing spaces in the input into account. The encoder-only model is both faster and better.
- We provide our whitespace correction models as a Python package with an easy-to-use command-line tool and as a web application; see Figure 1 and <https://whitespace-correction.cs.uni-freiburg.de>. The website provides links to public GitHub repositories with all our code, data, benchmarks, trained models, and a Docker setup. It also includes features to easily replicate our benchmark results and visualize model predictions on benchmarks.

2 Related Work

Recent work on spelling correction (Sakaguchi et al., 2017; Li et al., 2018; Pruthi et al., 2019; Jayanthi et al., 2020) and OCR postcorrection (Hämäläinen and Hengchen, 2019) predicts one

* Author contributions are stated in the end.



Legend: ✕ Clear input, 📄 Paste clipboard into input, 📁 Upload a text file, 📖 Choose an example input, 📚 Load a benchmark (see Section 4.1), ⏏ Insert whitespaces between all characters, ✂ Delete whitespaces between all characters, 📋 Copy output to clipboard, ⬇ Download output as text file, ▶ Run model on input, ✨ Live as-you-type whitespace correction¹

Figure 1: Our web interface for whitespace correction. The user can run any of our EO and ED models (panel for model selection not shown) on arbitrary text input. In the web app, there are tooltips instead of a legend.

word for every input token, without addressing space errors. To use these systems for text with a combination of spelling or OCR errors and whitespace errors, it is necessary to correct the spaces first. Other OCR postprocessing systems also try to correct space errors, but with limited success (Kissos and Dershowitz, 2016; D’hondt et al., 2017; Schulz and Kuhn, 2017; Nguyen et al., 2020). Bast et al. (2021) showed that space errors can be corrected separately with models that are robust against OCR and spelling errors.

Previous work on whitespace correction uses autoregressive models to determine the most likely segmentation of a given text into words. The used models are n -gram models, character-level recurrent neural network language models or character-level neural machine translation (NMT) models. Mikša et al. (2010) use a beam search with an n -gram language model to split unknown tokens in OCR’ed Croatian text into multiple words. Nastase and Hirschler (2018) use a character-level GRU NMT model to segment texts from the ACL anthology corpus. Soni et al. (2019) use n -gram statistics to split tokens in digitized historic English newspapers. Doval and Gómez-Rodríguez (2019) use a beam search with a character LSTM language model or n -gram language model for English word segmentation. Bast et al. (2021) use a beam search with a combination of a unidirectional character LSTM language model and a bidirectional LSTM whitespace classification model and introduce penalty terms to make use of existing spaces in the input text.

In contrast to previous work, our best approach does not use an autoregressive model, but instead addresses the task with a sequence-labeling classification model. We make use of the Transformer architecture, which improved the state of the art in many NLP tasks, including machine translation (Vaswani et al., 2017), language modeling (Radford et al., 2019), language understanding (Devlin et al., 2019), and Chinese word segmentation (Huang et al., 2020). We compare our approach with the previous best results from Bast et al. (2021) and other baselines.

3 Approach

We compare two approaches for the whitespace correction problem: A character-level encoder-decoder (ED) model and a byte-level encoder-only model (EO). Both models respect existing whitespace information in the input text. We pre-process the input text by removing duplicate, leading, and trailing whitespaces, and applying NFKC normalization².

3.1 Encoder-only (EO)

The EO approach treats the whitespace correction problem as a sequence-labeling task where we predict one of three repair tokens $\mathcal{R} = \{K, I, D\}$ for each character x_i in the input sequence

¹For live as-you-type whitespace correction we limit the input size to 512 characters, because we simply correct the full input after every keystroke. Non-live correction supports larger inputs such as text files in the order of several MB.

²This is a form of Unicode normalization. It is only relevant for our EO models with byte input, see Section 3.1

- $K \rightarrow$ Keep the character
- $I \rightarrow$ Insert a space before the character
- $D \rightarrow$ Delete the character

and use them afterwards to change the whitespacing in the input sequence accordingly.

For the EO approach, we directly input and embed UTF-8 encoded bytes instead of characters. We found the performance of using bytes directly to be on par with using characters, while introducing only a negligible runtime overhead. It also enables us to process any text without a character vocabulary or a special token for unknown characters. We also add sinusoidal positional encodings as defined by Vaswani et al. (2017) to the byte embeddings. To keep inference speeds high we aggregate all byte embeddings belonging to a character³ into a single embedding **before** processing them further⁴: Within a grapheme cluster we first average the byte embeddings belonging to individual code points, then average the code point embeddings to get the final character embedding.

To process the character embeddings, we employ a standard Transformer encoder (Vaswani et al., 2017) with a linear output layer on top, which allows us to predict a probability distribution over the repair tokens \mathcal{R} for each character in parallel. During inference we simply take the repair token with the highest probability for each character as output. However, we only allow the model to predict I between two non-space characters or D for space characters, otherwise we ignore the prediction and fall back to K :

$$y_i = \operatorname{argmax}_{r \in \mathcal{R}} p(r \mid x, i) \text{ with } 1 \leq i \leq n,$$

$$y_i \leftarrow \begin{cases} K & \text{if } y_i = D \text{ and } x_i \neq ' ' \\ K & \text{if } y_i = I \text{ and } (x_i = ' ' \text{ or } x_{i-1} = ' ') \\ y_i & \text{else.} \end{cases}$$

3.2 Encoder-decoder (ED)

For the ED approach, we tokenize the input text into a sequence of characters $x = (x_1, \dots, x_n)$ with $x_i \in \mathcal{C}$. \mathcal{C} is a character vocabulary containing all uppercase and lowercase letters of the English alphabet, common punctuation marks, and special tokens, e.g., for unknown characters.

³In accordance with the Unicode standard, we determine all characters in a UTF-8 encoded byte sequence using [grapheme cluster boundaries](#).

⁴For languages using non-Latin alphabets, e.g. Russian, where characters are usually encoded into multiple bytes in UTF-8 this can make a large difference.

The ED approach uses an encoder-decoder Transformer model (Vaswani et al., 2017) with a linear output layer trained to translate sequences of characters with space errors into sequences without space errors by outputting characters one by one. At each output step t , we use the ED decoder to predict a probability distribution over \mathcal{C} given the input sequence and the previous outputs $y_{<t} = (y_1, \dots, y_{t-1})$. To ensure that the ED model only changes the whitespaces of a sequence during inference, we limit the set of possible outputs at each step to the space character or the next character to copy from the input sequence x_j :

$$y_t = \operatorname{argmax}_{c \in \{ ' ', x_j \}} p(c \mid x, y_{<t}).$$

Sliding window Both the EO and ED approaches are trained with and limited to input sequences containing up to 512 tokens. Since real-world paragraphs often exceed this length bound, we use a sliding window approach during inference: We split input sequences into windows of 384 tokens and add the 64 tokens to the left and right of the window as additional context.⁵ For a given sequence we run our model on each individual window separately and recombine the whitespace correction results of all windows afterwards. For example, a 950 byte long sequence would be split into three consecutive windows $w_1 = (0, 448, 64)$, $w_2 = (64, 384, 64)$, and $w_3 = (64, 118, 0)$, specifying the sizes of the left context, the window itself, and the right context respectively. Note that at the beginning and end of the sequence, the left and right context sizes are 0, each window contains a maximum of 512 bytes, and all non-context sizes add up to 950.

We train a medium-sized and large model for each of the two approaches. All models use a hidden dimensionality of 512 and only differ in the number of encoder or decoder layers. See Table 1 for an overview over all models.

3.3 Training

We train our models on the publicly available training dataset from Bast et al. (2021).⁶ This dataset consists of 108,068,848 paragraphs extracted from

⁵We experimented with 16, 32, 64, and 128 as context size, but found it to have very little effect on correction quality. To be on the safe side we finally chose 64 as context size leaving us with 384 tokens for the actual window. Also, because of missing left and right context, the windows can contain more than 384 tokens at the beginning and end of sequences.

⁶At <https://whitespace-correction.cs.uni-freiburg.de>

Model	#Parameters	#Layers
ED _{medium}	22.2 M	3 encoder, 3 decoder
ED _{large}	44.2 M	6 encoder, 6 decoder
EO _{medium}	19.0 M	6 encoder
EO _{large}	38.0 M	12 encoder

Table 1: We choose the number of layers such that the corresponding ED and EO models have comparable numbers of parameters.

arXiv and Wikipedia articles. To generate pairs of correctly and misspelled sequences, the authors inject OCR and spelling errors artificially into the paragraphs using error models derived from text corpora, typo collections, and random character transformations. Additionally, we inject space errors into the paragraphs:

- In 10% of the paragraphs we remove all spaces.
- In 10% of the paragraphs we have a space between each pair of adjacent characters.
- In the remaining 80% of the paragraphs we insert a space between two adjacent non-space characters with probability 10% and delete an existing space with probability 20%.

The EO approach naturally suffers from an unbalanced class distribution, causing our models to reach a plateau during early stages of training where they predict K for all characters. To counteract that, we use a focal loss (Lin et al., 2017) with $\gamma = 2$. This causes our models to overcome the plateau by decreasing the influence of the dominant and (mostly) easy-to-predict K class. We also tried using a regular cross-entropy loss and weighing classes I and D higher, but found it to perform worse while also having one more hyperparameter. For the full training details see Appendix A.

4 Experiments

4.1 Benchmarks

We evaluate on a total of eight benchmarks (see Table 2 for an overview), with the first six coming from Bast et al. (2021):

- Three benchmarks with text from Wikipedia, one with whitespace errors only, one with whitespace and spelling errors, and one without spaces but with spelling errors. These benchmarks are called **Wiki**, **Wiki+**, and **Wiki+ no_␣** respectively.
- Two benchmarks based on text from arXiv with OCR errors and errors from PDF extraction, called **arXiv OCR** and **arXiv pdftotext**.

Benchmark	#Sequences	File size
Wiki	10,000	916 kB
Wiki+	10,000	916 kB
Wiki+no _␣	10,000	778 kB
arXiv OCR	10,000	1.4 MB
arXiv pdftotext	10,000	1.4 MB
ACL	500	83 kB
Doval	1000	82 kB
Runtime	3,500	396 kB

Table 2: A benchmark simply consists of two text files. The first file contains the corrupted input text where each line corresponds to a sequence with whitespace and spelling errors. Its size is shown in the file size column. The second text file then specifies the groundtruth sequences without whitespace errors accordingly.

- One benchmark based on sequences from the ACL anthology dataset containing OCR errors. This benchmark is called **ACL**.
- The word segmentation benchmark **Doval** from Doval and Gómez-Rodríguez (2019), with 1,000 sequences without any whitespace.
- A **Runtime** benchmark for measuring the inference runtimes of our models and baselines, built by randomly sampling 500 sequences from each of the seven benchmarks above.

4.2 Baselines

We reuse the following four baselines and their predictions from Bast et al. (2021):

- **Do nothing** This baseline keeps the input sequence unchanged. It is an interesting reference point for benchmarks with very few errors, like arXiv pdftotext.
- **Google** The authors copied erroneous sequences into a Google document⁷ and applied all suggested space edits.
- **Wordsegment** Wordsegment is a Python package for word segmentation.⁸ Before applying it to the text, all whitespaces are removed.
- **BID+** The best whitespace correction procedure from Bast et al. (2021), with the overall best hyperparameters (called *The One* in that paper). They perform a beam search with a combination of a unidirectional character-level LSTM language model and a bidirectional LSTM classification model to score whitespace insertions or deletions.

⁷At <https://docs.google.com>

⁸At <https://github.com/grantjenks/python-wordsegment>

In addition, we introduce an EO-like baseline called **ByT5**. Xue et al. (2022) released ByT5, a family of encoder-decoder Transformer models that input and output sequences of bytes and were pretrained on a masked token prediction task. We take the encoder of their smallest model (~217M parameters), add our byte-to-character aggregation scheme (see Section 3.1) and a linear output layer, and finetune it on 50M sequences from our training data for one epoch.

To our knowledge, the ByT5 models by Xue et al. (2022) are currently the only publicly available general purpose language models that work on character or byte level. Other openly accessible pretrained Transformer language models like the BERT (Devlin et al., 2019), T5 (Raffel et al., 2020), or OPT (Zhang et al., 2022) families are unsuitable for whitespace correction, because they all work with subword tokenization. Also, some of these are simply too large to permit a reasonable runtime and memory consumption.

4.3 Metric

Given two strings a and b that only differ in whitespaces, we define a function $\text{correction-ops}(a, b)$ that gives us the set of correction operations that we need to apply to turn a into b . A correction operation is a tuple $\langle r, i \rangle$ consisting of an insert or delete operation $r \in \{I, D\}$ and the character position i , $1 \leq i \leq |a|$ at which the operation has to be applied. Given a benchmark sample $\langle s, g, p \rangle$ as a tuple of an input sequence s , ground truth sequence g , and predicted sequence p , we define a F_1 -score as follows:

$$\begin{aligned} \mathcal{G} &= \text{correction-ops}(s, g) \\ \mathcal{P} &= \text{correction-ops}(s, p) \\ \text{TP} &= |\mathcal{G} \cap \mathcal{P}| \\ \text{FP} &= |\mathcal{P} \setminus \mathcal{G}| \\ \text{FN} &= |\mathcal{G} \setminus \mathcal{P}| \\ F_1 &= \begin{cases} 1 & \text{if } |\mathcal{G}| = |\mathcal{P}| = 0 \\ \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} & \text{else} \end{cases} \end{aligned}$$

For our evaluation metric we calculate the average F_1 -score over all benchmark samples. With this metric the Do nothing baseline gives the percentage of benchmark samples without space errors.

4.4 Results

Quality Table 3 shows the sequence-averaged F_1 -scores achieved by the baselines and our mod-

els on the 7 whitespace correction benchmarks.⁹ Compared to the neural models, Google and Wordsegment perform poorly, sometimes even worse than the Do nothing baseline.¹⁰ Our best models, ED_{large} and EO_{large} , perform on par with the best-so-far model BID+. In general, all four of our models achieve a high quality both in absolute terms as well as compared to the other baselines. We note, that EO_{large} performs slightly better than ED_{large} on 6 out of 7 benchmarks. A similar picture holds for the medium-sized variants. Comparing BID+ and EO_{large} , we see that the differences in F_1 -score between them are rather small ($\leq 0.6\%$) across all benchmarks. ByT5 achieves good quality overall, but falls behind both of our large and on some benchmarks even medium-sized models. We hypothesize that this is mainly due to it being pretrained on text without spelling and whitespace errors, which makes it hard to transfer its language modeling capabilities to this vastly different input distribution. Increasing the model size consistently leads to F_1 -score improvements across all benchmarks, both for the EO and the ED approach. See Appendix C for a showcase of some of the predictions and failure cases of our models on the benchmarks, and Appendix D for a visualization of attention maps.

Runtimes Table 4 shows runtime and GPU memory consumption on our Runtime benchmark (see Section 4.1). Wordsegment and BID+ use a batch size of 1 and cannot be easily modified to support larger batch sizes¹¹. We also show the performance of all other models for batch size 1. That way, we can see which improvements come from the approach, and which from an increased batch size. Appendix B provides results for more batch sizes for ByT5, ED, and EO.

As expected, the encoder-only models EO and ByT5 outperform BID+ and ED by a large margin,

⁹We additionally evaluated ChatGPT (gpt-3.5-turbo, available at <https://chat.openai.com>) on 100 random sequences from our benchmarks via the OpenAI API. ChatGPT usually changed more about the sequence than just the whitespace errors. Apart from that, we found the corrections to be of high quality, but the runtime to be very slow. ChatGPT took 668 seconds to correct the 100 sequences, which equals a throughput of 0.015 kB/s and is over 1,000 times slower than EO_{large} with batch size 1.

¹⁰For example, arXiv pdftotext contains only few errors, and it is hard to make few fixes at the right places and not introduce new errors.

¹¹In particular, BID+'s complicated decoding scheme including beam search caused its authors to implement inference only in an unbatched fashion.

Model	ACL	Wiki+ no_	Wiki+	Wiki	arXiv OCR	arXiv pdftotext	Doval
Do nothing	62.0	4.1	86.9	35.0	62.0	64.3	0.8
Google	75.6	16.7	91.9	76.0	83.9	86.1	-
Wordsegment	47.4	85.9	35.9	63.8	66.8	62.5	-
ByT5	87.1	98.7	95.7	98.2	96.5	94.8	99.4
BID+	87.8	99.0	98.0	98.8	97.6	95.5	99.9
ED _{medium}	86.0	98.9	96.5	98.5	96.7	95.4	99.5
ED _{large}	87.8	99.1	97.2	98.9	97.1	95.9	99.8
EO _{medium}	86.6	99.2	96.9	98.8	97.2	95.8	99.7
EO _{large}	87.5	99.3	97.6	99.0	97.3	96.1	99.9

Table 3: Sequence-averaged F₁-scores for our models and baselines on the whitespace correction benchmarks from Section 4.1. We show the best result for each benchmark in bold.

due to being able to correct each character in the input text simultaneously instead of one after another. They are even significantly faster than the (non-neural) Wordsegment baseline. Even with a batch size of 1, EO_{large} is over 75 times faster than the previous best model BID+. With a batch size of 128, EO_{large} achieves 213 kB/s, which is over 900 times faster than BID+.

Memory All of our models require relatively little memory and can easily be run on a standard GPU with reasonable batch sizes. We carried out additional tests with our EO_{large} model on a NVIDIA GeForce GTX 1080 Ti GPU, where we reached batch sizes of 305 and 399 for full precision and mixed precision inference, respectively, before getting OOM errors. We also determine a rough estimate of the amount of GPU memory required per sequence/batch element in Table 4. One can use these values to approximate the amount of GPU memory required for running a model with a certain batch size.

To investigate how performance scales with model size, we additionally trained another EO model called EO_{larger}, which has 18 layers and ~3.3 times more parameters than EO_{large}. EO_{larger} improves upon EO_{large} on every benchmark, but only by small margins with an average gain of 0.15 percentage points in sequence-averaged F₁-score. Compared with EO_{large}, it also requires about three times more memory and runs only at 60% of its speed. Here, the tradeoff between quality improvement, speed loss, and increase in memory consumption seems less favorable than the one between EO_{large} and EO_{medium}. We leave it for future work

to investigate how much performance can improve further by using even larger models.

5 Demo

We provide open access to all of the EO and ED models in form of a Python package which can be installed via pip¹² or from source. The package comes with a command line tool and a Python API both of which enable the user to correct whitespace errors in arbitrary text. Additionally, the command line tool provides an option for running a whitespace correction JSON API and thereby enables access to our models not only from Python code but also from other programming and development environments.

In addition to the Python package, we also provide a web interface at <https://whitespace-correction.cs.uni-freiburg.de> that allows users to correct whitespaces in arbitrary text; see Figure 1. In particular, the web app allows to evaluate the output against an error-free ground truth and provides quick access to all of the used benchmarks. Therefore, the easiest way to reproduce the results of our models on all benchmarks as presented in this paper is through the web interface.

For more information visit our GitHub repositories at <https://github.com/ad-freiburg/whitespace-correction> and <https://github.com/ad-freiburg/text-correction-benchmarks>.

6 Conclusion

We have shown that carefully trained encoder-only Transformers perform whitespace correction with

¹²At <https://pypi.org/project/whitespace-correction>

Model	Batch size	Runtime seconds	Throughput sequences / sec	Throughput kB / sec	GPU memory MiB	GPU memory MiB / sequence
Wordsegment	1	246.1	14.22	1.6	-	-
ByT5	1	31.9	109.6	12.7	974	~51
BID+	1	1,725	2.0	0.2	-	-
ED _{medium}	1	1,402	2.5	0.3	144	~36
ED _{large}	1	2,432	1.4	0.2	234	~36
EO _{medium}	1	13.8	253.9	29.4	118	~28
EO _{large}	1	22.5	155.4	18.0	184	~28
ByT5	128	4.2	833.6	96.4	7,402	~51
ED _{large}	128	151.8	23.1	2.7	4,832	~36
EO _{large}	128	1.9	1,842	213.0	3,704	~28

Table 4: Inference runtimes on our Runtime benchmark (see Section 4.1) on a NVIDIA A100 GPU and an Intel Xeon Platinum 8358 CPU with mixed precision enabled. Wordsegment and BID+ only support a batch size of 1. Results for more batch sizes for ByT5, ED, and EO can be found in Appendix B.

the same high quality as the best previous work, but over 900 times faster. A classical encoder-decoder Transformer can achieve the same quality, but with little to no gains in runtime speed. Our software is open source and accessible as a Python package as well as via a dedicated website.

A logical next step in this line of work is to combine whitespace correction with spelling correction. Recent large language models like GPT-3 (Brown et al., 2020) can solve both tasks at once with high quality, but they are slow and expensive to run due to their size and autoregressive decoder architecture. By separating the tasks, as advocated by Bast et al. (2021), each can be solved efficiently with specialized models, like our EO models for whitespace correction. It remains an interesting open question whether a single model can achieve both high quality and reasonably cheap inference.

Due to working directly with bytes, our EO models could be trained and applied across multiple languages. However, because datasets and benchmarks for whitespace correction in non-English languages are yet to be created, we leave the development of multilingual models for future work.

Author contributions

Most of the work behind this paper was done by S.W., with M.H. and H.B. acting as advisers. In particular, S.W. did all of the implementation work and the evaluation. All authors wrote the paper, with S.W. doing the largest part.

Acknowledgement

The authors acknowledge support by the state of Baden-Württemberg through bwHPC. M.H. is funded by the Helmholtz Association’s Initiative and Networking Fund through Helmholtz AI.

Ethics Statement

Our models are trained on a mix of Wikipedia and arXiv articles, so they are naturally exposed to the biases represented in that data. However, since the models are restricted to only changing the whitespacing in text either by design (for the EO models) or by the decoding procedure (for the ED models), we consider the ethical concerns of using our models to be small.

References

- Hannah Bast, Matthias Hertel, and Mostafa M. Mohamed. 2021. [Tokenization repair in the presence of spelling errors](#). In *Conference on Computational Natural Language Learning (CoNLL ’21)*, pages 279–289.
- 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.
- 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 '19)*, pages 4171–4186.
- Eva D'hondt, Cyril Grouin, and Brigitte Grau. 2017. [Generating a training corpus for OCR post-correction using encoder-decoder model](#). In *International Joint Conference on Natural Language Processing (IJCNLP '17)*, pages 1006–1014.
- Yerai Doval and Carlos Gómez-Rodríguez. 2019. [Comparing neural- and N-gram-based language models for word segmentation](#). *Journal of the Association for Information Science and Technology*, 70(2):187–197.
- Mika Härmäläinen and Simon Hengchen. 2019. [From the paft to the fiiture: A fully automatic NMT and word embeddings method for OCR post-correction](#). In *International Conference on Recent Advances in Natural Language Processing (RANLP '19)*, pages 431–436.
- Weipeng Huang, Xingyi Cheng, Kunlong Chen, Taifeng Wang, and Wei Chu. 2020. [Towards fast and accurate neural chinese word segmentation with multi-criteria learning](#). In *International Conference on Computational Linguistics (COLING '20)*, pages 2062–2072.
- Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. [NeuSpell: A neural spelling correction toolkit](#). In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP Demos '20)*, pages 158–164.
- Ido Kissos and Nachum Dershowitz. 2016. [OCR error correction using character correction and feature-based word classification](#). In *International Workshop on Document Analysis Systems (DAS '16)*, pages 198–203.
- Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. 2018. [Spelling error correction using a nested RNN model and pseudo training data](#). *CoRR*, abs/1811.00238.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. [Focal loss for dense object detection](#). In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations (ICLR '19)*.
- Mladen Mikša, Jan Šnajder, and Bojana Dalbelo Bašić. 2010. [Correcting word merge errors in croatian texts](#). *International Conference on Formal Approaches to South Slavic and Balkan Languages (FASSBL '10)*.
- Vivi Nastase and Julian Hitschler. 2018. [Correction of OCR word segmentation errors in articles from the ACL collection through neural machine translation methods](#). In *International Conference on Language Resources and Evaluation (LREC '18)*.
- Thi-Tuyet-Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickaël Coustaty, and Antoine Doucet. 2020. [Neural machine translation with BERT for post-OCR error detection and correction](#). In *Joint Conference on Digital Libraries (JCDL '20)*, pages 333–336.
- Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. [Combating adversarial misspellings with robust word recognition](#). In *Conference of the Association for Computational Linguistics (ACL '19)*, pages 5582–5591.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 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(140):1–67.
- Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. [Robsut wrod reocginiton via semi-character recurrent neural network](#). In *Conference on Artificial Intelligence (AAAI '17)*, pages 3281–3287.
- Sarah Schulz and Jonas Kuhn. 2017. [Multi-modular domain-tailored OCR post-correction](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP '17)*, pages 2716–2726.
- Sandeep Soni, Lauren F. Klein, and Jacob Eisenstein. 2019. [Correcting whitespace errors in digitized historical texts](#). In *Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH '19)*, pages 98–103.
- 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 (NIPS '17)*, pages 5998–6008.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. [ByT5: Towards a token-free future with pre-trained byte-to-byte models](#). *Transactions of the Association for Computational Linguistics*, 10:291–306.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. [Opt: Open pre-trained transformer language models](#).

A Training setup

We train all our medium and large sized models for three epochs over the full training data with a maximum sequence length of 512 and up to 65,536 tokens per batch. This amounts to about 1.2M training steps. We warmup the learning rate linearly to 10^{-4} over the first percent of training steps and decay it afterwards towards zero using a cosine schedule. We use AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a weight decay of 0.01, and clip gradients to a norm of 1. We also use a dropout rate of 0.1 throughout our models. Finally, we keep the model checkpoint corresponding to the lowest validation loss as our final model. Training takes about 4-5 days for all of our models on a single NVIDIA V100/A100 GPU.

B Batched runtimes

Model	Batch size	Runtime seconds	Throughput sequences / sec	Throughput kB / sec	GPU memory MiB	GPU memory MiB / sequence
ByT5	16	5.0	696.2	80.5	1,732	~51
	128	4.2	833.6	96.4	7,402	
ED _{medium}	16	174.4	20.1	2.3	686	~36
	128	106.9	32.7	3.8	4,744	
ED _{large}	16	264.03	13.2	1.5	774	~36
	128	151.8	23.1	2.7	4,832	
EO _{medium}	16	2.2	1,596	184.5	550	~28
	128	1.6	2,143	247.7	3,618	
EO _{large}	16	2.7	1,277	147.7	620	~28
	128	1.9	1,842	213.0	3,704	

The table above shows runtimes using batched inference on our Runtime benchmark (see Section 4.1) on a NVIDIA A100 GPU and an Intel Xeon Platinum 8358 CPU with mixed precision enabled. For each model we report both batch size 16 and 128. We only show the models from Table 4 that support batched inference in their implementation. The EO models reach a throughput of well over 200 kB/s with a batch size of 128 while requiring less than 4GB of GPU memory. We consider them to be fast and memory efficient enough to be used in practical applications, even on less powerful end-user devices like laptops.

C Sample predictions and failure cases

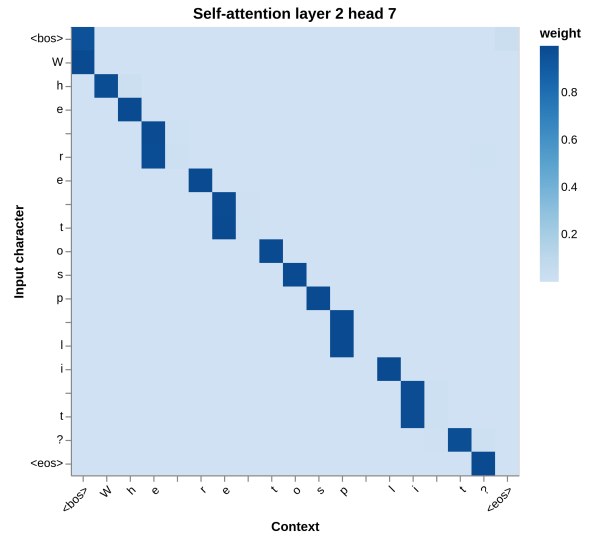
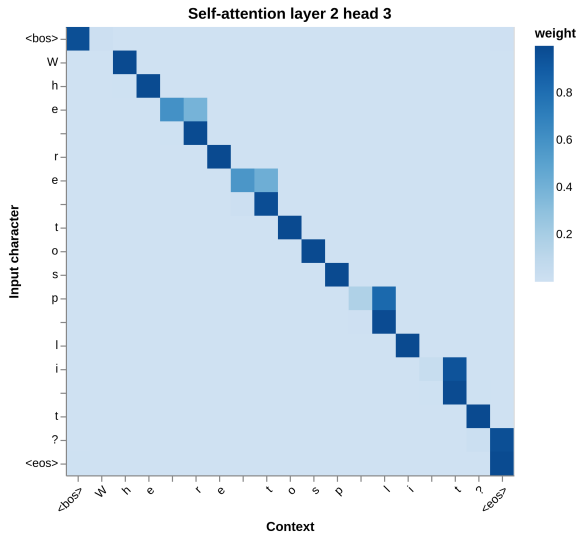
We present some predictions and failure cases of our EO_{large} model on selected benchmark sequences in the following. In accordance with our website at <https://whitespace-correction.cs.uni-freiburg.de> we mark correct changes (true positives) in the input text with green, incorrect changes (false positives) with red, and missing changes (false negatives) orange. See our website if you want to evaluate and visualize our models' predictions on your own texts. Marked whitespaces are shown as # for visualization purposes.

Input	Prediction	Comment
Unlike in Canada, the American States are responsible for the organisation of federal elections in the United States.	Unlike#in#Canada.#the#American#States#are#responsible#for#the#organisation#of#federal#elections#in#the#United#States.	For sentences that contain no spelling errors and no exotic words, our model is almost always able to perfectly correct the sequence, even if it contains no whitespaces at all.
The Exit Players have trained with members of Improvised Shakespeare, Paralellogramophonograph, andi Baby Wants Candy, os well as instructors from the Groundlings, the PIT, iO, and Coldtowne Theater.	The Exit Players have trained with members of Improvised Shakespeare, Paralellogramo#phonograph, andi Baby Wants Candy, os well as instructors from the Groundlings, the PIT, iO, and Coldtowne Theater.	Complex composite words or proper names are sometimes split or merged by our model. To correctly identify them one requires either domain knowledge or very good language understanding.
He has also played league chess in the Chess Bundesliga, for Porz an Werder Bremen.	He has also played league chess in the Chess Bundesliga, for Porz#an Werder Bremen.	Similar to the sample above. Our model predicts that <i>Porzan Werder Bremen</i> is what the chess club is called, but actually this should have been <i>Porz and Werder Bremen</i> , where Porz is a German city near Cologne.
Brian Catling (born 1948 in London) is an English sculptor, poet, novelist, film maker and perofmrance artist.	Brian Catling (born 1948 in London) is an English sculptor, poet, novelist, film#maker and perofmrance artist.	Here our model merges the words <i>film</i> and <i>maker</i> , which according to the Cambridge dictionary is also a valid English word. A reminder that not all benchmark ground truths are unambiguous.
X stems from Y X eases Y *Y results in X Y is related to X *X is result of Y X is linked to Y	X stems from Y X eases Y *Y results in X Y is related to X *X is result of Y X is linked to Y	Here we would expect our model to either insert a whitespace before both Y and X, or remove the whitespace before * in both cases. Instead it keeps the sequence unchanged.
Ju ly l ducation Programs Beginning after J a n ~ a r y 1, 1976 Roger R o s e ~ b l a t t , Divi-sion Director -202-382-5891 Procjrhm grants for c r i t i c a l re-examination of t h e content, o r g a h i z a t i o n , and method of presenta tion of a group of related courses or an ordered program of study in the humanities. The central topic can be a region, culture, era, etc.; o r a program can be defined by a cur r - i c u l a r level. L i m i t , \$ 1 8 0 , 0 0 0 i n three years.	July l ducation Programs Beginning after Ja#n#~#a#ry 1, 1976 Roger Rose#~#b#l#a#t#t, Divi-sion Director -202-382-5891 Procjrhm grants for critical re-examination of the content, orgahization , and method of presentation of a group of related courses or an ordered program of study in the humanities. The central topic can be a region, culture, era, etc.; or a program can be defined by a curri-cular level. Limit , \$180,000 in three years.	Exotic characters within words which are not often seen during training can cause or model to not be able to correctly merge the full word. Here the model missed to correctly predict the words <i>Jan~ary</i> and <i>Rose~blatt</i> .

D Attention visualization

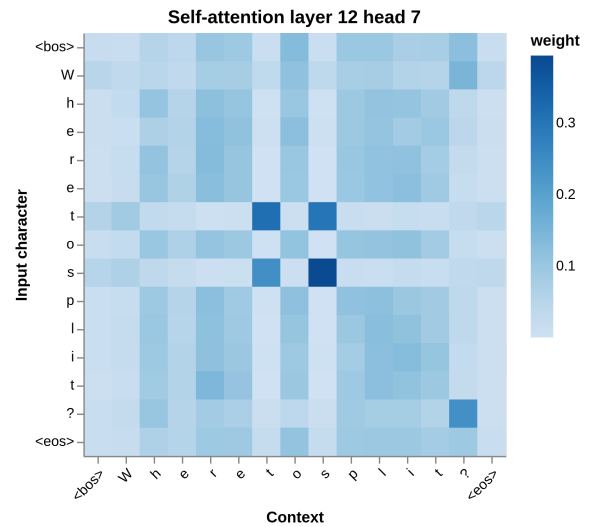
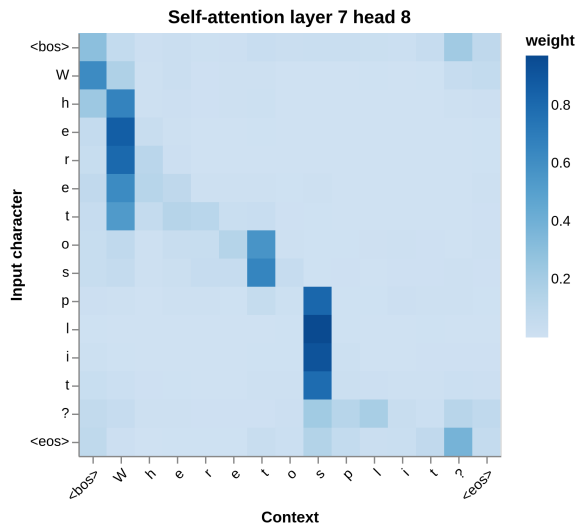
We look at the self-attention maps produced during the forward pass of our EO models. In earlier layers we mostly find local or character-specific attention patterns, while in the middle and later layers our models ultimately seem to learn to identify word boundaries.

All of the following self-attention maps are normalized row-wise, meaning each row displays the attention distribution for the input character on its left over all input characters, which we call context in the figures.



(a) Attention head in layer 2/12 looking at the following non-whitespace character.
Input text is *Where to split?*

(b) Attention head in layer 2/12 looking at the previous non-whitespace character.
Input text is *Where to split?*



(c) Attention head in layer 7/12 identifying the individual words in the input text (indicated by the vertical bars).
Input text is *Wheretosplit?*

(d) Attention head in layer 12/12 marking where whitespaces should be inserted into the input text.
Input text is *Wheretosplit?*

Exemplary attention maps of selected heads produced in early, middle, and late layers while running our EO_{large} model. Input texts are deliberately chosen make the attention patterns as clear as possible.

ESPnet-ST-v2: Multipurpose Spoken Language Translation Toolkit

Brian Yan*¹ Jiatong Shi*¹ Yun Tang² Hirofumi Inaguma²
Yifan Peng¹ Siddharth Dalmia¹ Peter Polák³ Patrick Fernandes¹
Dan Berrebbi¹ Tomoki Hayashi⁴ Xiaohui Zhang² Zhaoheng Ni²
Moto Hira² Soumi Maiti¹ Juan Pino² Shinji Watanabe^{1,5}

¹Carnegie Mellon University ²Meta AI

³Charles University ⁴Nagoya University ⁵Johns Hopkins University

{byan, jiatongs}@cs.cmu.edu

Abstract

ESPnet-ST-v2 is a revamp of the open-source ESPnet-ST toolkit necessitated by the broadening interests of the spoken language translation community. ESPnet-ST-v2 supports 1) offline speech-to-text translation (ST), 2) simultaneous speech-to-text translation (SST), and 3) offline speech-to-speech translation (S2ST) – each task is supported with a wide variety of approaches, differentiating ESPnet-ST-v2 from other open source spoken language translation toolkits. This toolkit offers state-of-the-art architectures such as transducers, hybrid CTC/attention, multi-decoders with searchable intermediates, time-synchronous blockwise CTC/attention, Translatotron models, and direct discrete unit models. In this paper, we describe the overall design, example models for each task, and performance benchmarking behind ESPnet-ST-v2, which is publicly available at <https://github.com/espnet/espnet>.¹

1 Introduction

The objective of this project is to contribute to the diversity of the open-source spoken language translation ecosystem. Toward this, we launched this ESPnet-ST-v2 update in collaboration with researchers working on Fairseq (Ott et al., 2019) and TorchAudio (Yang et al., 2021b). This project focuses on: offline speech-to-text (ST), simultaneous speech-to-text (SST), and offline speech-to-speech (S2ST). These three spoken language translation tasks have drawn significant interest, as evidenced by rising IWSLT² shared task participation.

The ST task can be considered a base form of spoken language translation. Early approaches to ST stemmed from coupling statistical automatic speech recognition (ASR) (Huang et al., 2014) and text-to-text translation (MT) (Al-Onaizan et al., 1999), and this type of cascaded approach is still

common in the neural network era (Bentivogli et al., 2021; Zhang et al., 2022). End-to-end differentiable (E2E) approaches have recently emerged as an alternative offering greater simplicity and superior performance in some cases (Inaguma et al., 2021b); however, E2E approaches still benefit from techniques originating from ASR and MT (Gaido et al., 2021; Inaguma et al., 2021a).

SST modifies ST by imposing an additional streaming requirement, where systems are expected to produce textual outputs while incrementally ingesting speech input. Both the aforementioned cascaded and end-to-end approaches to ST have been adapted for SST (Ma et al., 2020b; Iranzo-Sánchez et al., 2021; Chen et al., 2021), although the more direct nature of the latter may be advantageous for latency-sensitive applications. On the other hand, S2ST extends ST by producing target speech rather than target text. Again, cascaded approaches of ST followed by text-to-speech (TTS) came first (Waibel et al., 1991; Black et al., 2002) and E2E approaches followed (Jia et al., 2019; Lee et al., 2022a; Jia et al., 2022a; Inaguma et al., 2022), with the latter offering smaller footprints and greater potential to retain source speech characteristics.

Given the recent swell in E2E ST, SST, and S2ST research, we have revamped ESPnet-ST (Inaguma et al., 2020) which previously only supported E2E ST. In particular, this work:

- Implements ST, SST, and S2ST using common Pytorch-based modules, including encoders, decoders, loss functions, search algorithms, and self-supervised representations.
- Builds a variety of example E2E models: attentional encoder-decoders, CTC/attention, multi-decoders with searchable intermediates, and transducers for ST. Blockwise attentional encoder-decoders, time-synchronous blockwise CTC/attention and blockwise transducers for SST. Spectral models (i.e. Translatotron) and

¹Please see our documentation for [ST/SST](#) and [S2ST](#) to get started. Example models and tutorials are provided.

²International Workshop on Spoken Language Translation

discrete unit based models for S2ST.

- Benchmarks the ST, SST, and S2ST performance of ESPnet-ST-v2 against top IWSLT shared task systems and other prior works.

With this major update, ESPnet-ST-v2 keeps pace with the interests of the community and offers a variety of unique features, making it a valuable complement to Fairseq (Wang et al., 2020), NeurST (Zhao et al., 2021), and other spoken language translation toolkits.

2 Related Works

ESPnet-ST-v2 follows a long line of open-source speech processing toolkits which can support spoken language translation (Zenkel et al., 2018; Shen et al., 2019; Kuchaiev et al., 2019; Hayashi et al., 2020; Wang et al., 2020; Zhao et al., 2021).

In Table 1 we compare ESPnet-ST-v2 to Fairseq (Wang et al., 2020) and NeurST (Zhao et al., 2021), two toolkits which also cover multiple types of spoken language translation. Fairseq and NeurST offer cascaded and E2E approaches to ST and SST (some of which are not offered by ESPnet-ST-v2). Meanwhile, ESPnet-ST-v2 focuses on E2E approaches and offers multiple unique core architectures not covered by the other toolkits. For S2ST, Fairseq and ESPnet-ST-v2 both offer a range of approaches. All told, ESPnet-ST-v2 offers the greatest variety across ST, SST, and S2ST – however, we view these toolkits as complementary. The following section elaborates on the unique features of ESPnet-ST-v2.

3 ESPnet-ST-v2

In this section, we first describe the overall design and then introduce a few key features.

3.1 Modular Design

Figure 1 illustrates the software architecture of ESPnet-ST-v2. This modular design is an improvement over the ESPnet-ST-v1 where monolithic model and task definitions made it more difficult to extend and modify the toolkit. We also designed ESPnet-ST-v2 such that modules developed for adjacent tasks (e.g. ASR, TTS, MT) can also be readily used for spoken language translation.

In ESPnet-ST-v2 major neural network modules, such as frontends, encoders, decoders, search, and loss functions, inherit from common abstract classes making them easy to interchange. These modules, which are detailed further in the next

FEATURES	ESPNET-ST-v2	ESPNET-ST-v1	FAIRSEQ-S2T	NEURST
Offline ST	✓	✓	✓	✓
End-to-End Architecture(s)	✓	✓	✓	✓
Attentional Enc-Dec	✓	✓	✓	✓
CTC/Attention	✓	-	-	-
Transducer	✓	-	-	-
Hierarchical Encoders	✓	-	-	-
Multi-Decoder	✓	-	-	-
Cascaded Architectures	✓	✓	✓	✓
Speech SSL Representations	✓ ¹	-	✓	-
Speech & Text Pre-training	✓	✓	✓	✓
Joint Speech/Text Pre-training	-	-	✓	-
Simultaneous ST	✓	-	✓	✓ ³
End-to-End Architecture(s)	✓	-	✓	-
Contextual Block Encoders	✓	-	-	-
Blockwise Attn Enc-Dec	✓	-	-	-
Blockwise CTC/Attention	✓	-	-	-
Blockwise Transducer	✓	-	-	-
Wait-K Attn Enc-Dec	-	-	✓	-
Monotonic Attn Enc-Dec	-	-	✓	-
Cascaded Architectures	-	-	✓	✓ ³
Offline S2ST	✓	-	✓	-
End-to-End Architecture(s)	✓	-	✓	-
Spec Enc-Dec (Translatotron)	✓	-	✓	-
Spec Multi-Dec (Translatotron 2)	✓	-	✓	-
Discrete Enc-Dec (Speech-to-Unit)	✓	-	✓	-
Discrete Multi-Decoder (UnitY)	✓	-	✓	-
Speech SSL Representations	✓ ¹	-	✓	-
Neural Vocoder Support	✓ ²	✓	✓	-

Table 1: Key features of ESPnet-ST-v2 compared to ESPnet-ST-v1 (Inaguma et al., 2020), Fairseq (Wang et al., 2020), and NeurST (Zhao et al., 2021). Comparison intends to highlight unique features of ESPnet-ST-v2 and not to comprehensively review all toolkits. ¹Supports S3PRL (Yang et al., 2021a). ²Supports both spectral and discrete. ³Only supports text-to-text.

subsection, are used as building blocks in wrapper classes which are used to construct model architectures. Then the fully constructed models are fed to task wrappers which prepare data loaders, initialize models, and handle training/validation. For inference, pythonic APIs invoke search algorithms over the trained models and direct outputs to scoring scripts. For instance, the third-party SimulEval tool for evaluating SST latency (Ma et al., 2020a) is integrated via this API layer. We are also integrating with TorchAudio (Yang et al., 2021b) in the same manner. Finally, recipe scripts define experimental pipelines from data preparation to evaluation.

3.2 Key Features

Each of the following modeling components feature a variety of interchangeable approaches.

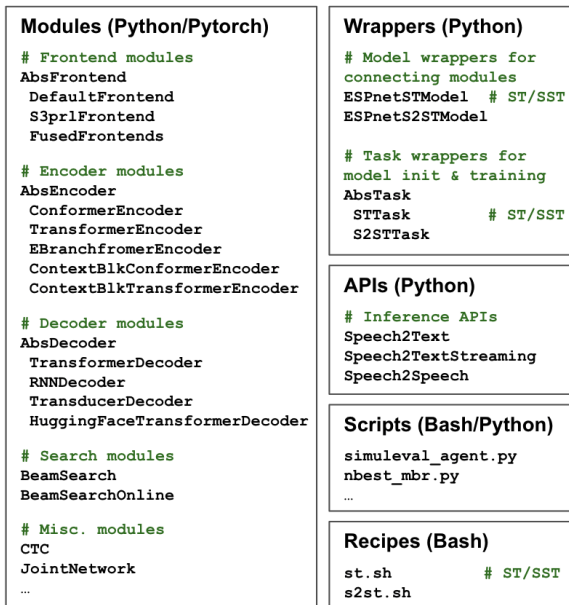


Figure 1: Software architecture of ESPnet-ST-v2.

Frontends & Targets Spectral features (e.g. FBANK) and features extracted from speech self-supervised learning (SSL) representations are supported, as well as fusions over multiple features (Berrebbi et al., 2022). For speech SSL features, ESPnet-ST-v2 integrates with the S3PRL toolkit (Yang et al., 2021a). These speech SSL representations are also used to generate discrete targets for S2ST (Lee et al., 2022a).

Encoder Architectures Conformer (Gulati et al., 2020; Guo et al., 2021), Branchformer (Peng et al., 2022), EBranchformer (Kim et al., 2023), and Transformer (Vaswani et al., 2017; Karita et al., 2019) encoder architectures are supported for ST and S2ST. For SST, a blockwise scheme is adopted following (Tsunoo et al., 2021; Deng et al., 2022) to form contextual block Conformer and Transformer encoders. Intermediate CTC (Lee and Watanabe, 2021) and Hierarchical CTC (Sanabria and Metze, 2018) encoding are also supported; these techniques have been shown to stabilize deep encoder optimization (Lee and Watanabe, 2021) and improve representations for sequence tasks involving source-to-target re-ordering (Yan et al., 2023).

Decoder Architectures Attentional Transformer and recurrent neural network decoders are supported (Karita et al., 2019). Multi-decoder schemes which allow for E2E differentiable decoder cascades via searchable hidden intermediates (Dalmia et al., 2021), are also supported; this technique

has been shown to improve sequence modeling for tasks which naturally decompose into sub-tasks. Finally, large language model decoders (e.g. mBART (Liu et al., 2020b)) can be adopted through an integration with HuggingFace (Wolf et al., 2020).

Loss Functions Cross-entropy (for attentional decoders), CTC, and Transducer are supported for ST and SST. Multi-objective training with CTC/attention and CTC/transducer as well as multi-task training (e.g. ASR/MT/ST) is also supported. For S2ST, L1 and mean square error losses are also supported for spectral models.

Search Algorithms For offline attentional decoder models, label-synchronous beam search is supported with optional CTC joint decoding for multi-objective models (Watanabe et al., 2017). For offline Transducer models, the original Graves beam search (Graves, 2012) as well as time-synchronous and alignment-synchronous beam search (Saon et al., 2020) beam searches are supported. For SST, both incremental decoding and non-incremental (allowing re-translation) decoding are supported (Liu et al., 2020a). Blockwise attentional decoder models use a label-synchronous beam search or time-synchronous beam search if a CTC branch is available. Blockwise transducer models use time-synchronous beam search.

Synthesis & Post-processing For ST, Minimum Bayes Risk (MBR) ensembling (Fernandes et al., 2022) is supported for leveraging quality-metrics (e.g. BLEU) to compare and rank n-best outputs from one or more models. For S2ST, neural vocoders are supported for both spectral and discrete inputs (Hayashi et al., 2020, 2021).

4 Example Models

In this section, we introduce example models which are pre-built in ESPnet-ST-v2 using the neural network components described in the previous section. These examples include state-of-the-art core architectures, as evidenced by prior studies and our performance benchmarking (presented in §5).

4.1 ST Models

CTC/Attention (CA) Following Yan et al. (2023), we use Conformer encoders with hierarchical CTC encoding and Transformer decoders. The hierarchical CTC encoding, which aligns the first N layers of the encoder towards ASR targets and the last M layers towards ST targets, regularizes

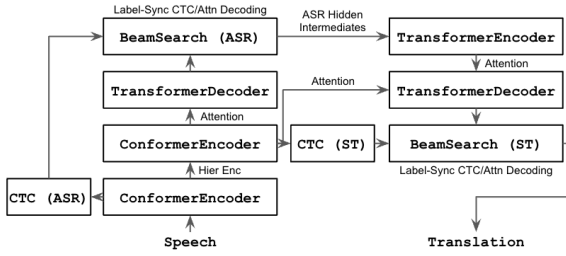


Figure 2: Multi-Decoder CTC/Attention for ST.

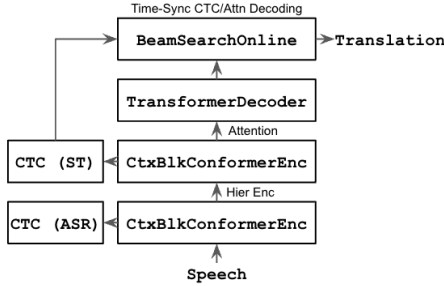


Figure 3: Time-Sync Blockwise CTC/Attn for SST.

the final encoder representations to be monotonic with respect to the target. CTC/attention models are jointly decoded using either label-synchronous (wherein the attention branch is primary) or time-synchronous (wherein the CTC branch is primary) beam search. For offline tasks, label-synchrony has shown greater performance (Yan et al., 2023).

Multi-Decoder CTC/Attention (MCA) As shown in Figure 2, the Multi-decoder decomposes ST into two sub-tasks, logically corresponding to ASR and MT encoder-decoder models, while maintaining E2E differentiability (Dalmia et al., 2021). This Multi-decoder scheme is also combined with the CTC/attention scheme described in the blurb above, following Yan et al. (2022). We use Conformer encoders with hierarchical CTC for encoding speech and Transformer encoders for encoding intermediate ASR text. We use Transformer decoders for both ASR and ST. During inference, the ASR stage is decoded first and then the final MT/ST stage is decoded; both stages use label-synchronous joint CTC/attention beam search.

4.2 SST Models

Time-Synchronous Blockwise CTC/Attention (TBCA) As shown in Figure 3, we adapt the aforementioned CTC/attention model for ST (§4.1) to SST by replacing the Conformer encoder with a contextual block Conformer (Tsunoo et al.,

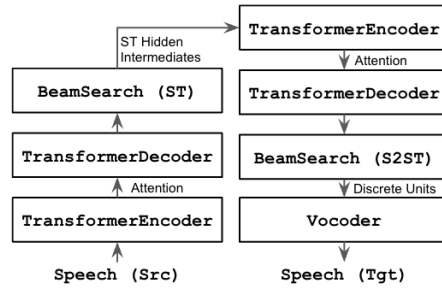


Figure 4: Discrete Multi-Decoder (UnitY) for S2ST.

2021). During inference, we initially followed Deng et al. (2022) and used the label-synchronous CTC/attention beam search originally proposed for ASR by Tsunoo et al. (2021). However, we found that label-synchrony results in overly conservative boundary block detection for SST. Therefore we opt instead for the time-synchronous variant which relies on CTC’s more robust end-detection (Yan et al., 2023) to control boundary block detection; this change reduces latency without sacrificing quality. To perform incremental decoding without re-translation (as expected by SimulEval), hypotheses are pruned after processing all of the time steps for each encoder block.

Blockwise Transducer (BT) As demonstrated by Xue et al. (2022), Transducers can be effectively applied to SST despite the monotonic nature of their underlying alignment model. We build Transducers for SST using contextual block Conformer encoders and unidirectional LSTM decoders. We found that the aforementioned hierarchical CTC encoding (§4.1) improves training stability and convergence rate. During inference, we found that the time-synchronous algorithm described by Saon et al. (2020) outperformed the original Graves decoding (Graves, 2012) and the later proposed alignment-synchronous algorithms (Saon et al., 2020). We also found that length normalization is required to avoid overly short outputs. Incremental decoding is applied in the same manner as for TBCA.

4.3 S2ST Models

Spectral Multi-Decoder (Translatotron 2) Similar to the MCA model for ST (§4.1), the spectral Multi-decoder (Jia et al., 2022a) decomposes S2ST into ST and TTS sub-tasks. The ST sub-task is modeled with an encoder-decoder network while the TTS sub-task is modeled with an auto-

TOOLKIT	MODEL TYPE	DE	ES	FR	avg
OFFLINE SPEECH TRANSLATION (ST)		BLEU \uparrow			
NeurST (Zhao et al., 2021)	Attentional Enc-Dec (AED)	22.8	27.4	33.3	27.8
Fairseq (Wang et al., 2020)	Attentional Enc-Dec (AED)	22.7	27.2	32.9	27.6
ESPnet-ST-v1 (Inaguma et al., 2020)	Attentional Enc-Dec (AED)	22.9	28.0	32.8	27.9
ESPnet-ST-v2 (this work)	Multi-Decoder CTC/Attn (MCA)	27.9	32.1	38.5	32.8
SIMULTANEOUS SPEECH TRANSLATION (SST)		BLEU \uparrow / AL \downarrow			
Fairseq (Wang et al., 2020)	Wait-K Attentional Enc-Dec (WAED)	18.6 / 6.8	22.9 / 6.9	28.5 / 6.7	23.3 / 6.8
ESPnet-ST-v2 (this work)	Time-Sync Blockwise CTC/Attn (TBCA)	23.5 / 2.3	29.2 / 2.4	32.7 / 2.3	28.5 / 2.3
OFFLINE SPEECH-TO-SPEECH TRANSLATION (S2ST)		ASR-BLEU \uparrow			
Fairseq (Inaguma et al., 2022)	Discrete Multi-Decoder (UnitY)	25.5	32.3	30.9	29.6
ESPnet-ST-v2 (this work)	Discrete Multi-Decoder (UnitY)	23.7	32.0	33.1	29.6

Table 2: Overview of ESPnet-ST-v2’s ST, SST, and S2ST performances compared to other open-source toolkits. Results are presented on MuST-C-v1 (English-to-X) for ST/SST and on CVSS-C (X-to-English) for S2ST.

regressive synthesizer. The synthesizer attends over both the ST-encoder and ST-decoder hidden states. We use Transformers for the ST encoder-decoder and a Tacotron-style (Wang et al., 2017) decoder as the synthesizer. During inference, we first use beam search for the ST sub-task and then autoregressively generate Mel-spectrograms. The final waveform speech is generated with a HiFi-GAN vocoder (Kong et al., 2020).

Discrete Multi-Decoder (UnitY) The UnitY model (Inaguma et al., 2022) is similar to Translatotron 2, but critically predicts discrete units of speech SSL representations rather than spectral information in the final stage. In other words, UnitY is Multi-decoder consisting of a ST sub-task followed by a text-to-unit (T2U) sub-task (see Figure 4). We use Transformer-based encoder-decoders for both sub-tasks. During inference, the ST stage is first decoded and then followed by the T2U stage. Both stages use label synchronous beam search. The final speech is generated with a unit HiFi-GAN vocoder with FastSpeech-like duration prediction (Polyak et al., 2021; Lee et al., 2022a), which is separately trained in the Parallel-WaveGAN toolkit (Hayashi et al., 2020, 2021).

5 Performance Benchmarking

In this section, we 1) compare open-source toolkits 2) compare our different example models and 3) compare our models with top IWSLT shared task systems and state-of-the-art prior works.

5.1 Experimental Setup

Please refer to §A.1 for reproducibility details. The following is only a summary of our setup.

MODEL	HIERENC	BLEU \uparrow
Attn Enc-Dec (AED)	-	25.7
Multi-Decoder Attn Enc-Dec (MAED)	-	27.6
CTC/Attention (CA)	✓	28.6
Multi-Decoder CTC/Attn (MCA)	✓	28.8
Transducer (T)	✓	27.6

Table 3: Example ST models – results on MuST-C-v2 En-De tst-COMMON. HierEnc=Hierarchical Encoder.

Data We use MuST-C-v1 or MuST-C-v2 (Di Gangi et al., 2019) for ST/SST and CVSS-C for S2ST (Jia et al., 2022b). For IWSLT comparisons, we combine MuST-C-v1, MuST-C-v2, and ST-TED (Niehues et al., 2018) for ST/SST.

Models Unless otherwise indicated, we use a "base" setting for our models. Our base models have 40-80M trainable parameters across all tasks and are trained on a \sim 400h of single language pair data from a single corpus. For ST/SST, we also use a "large" setting for benchmarking against IWSLT submissions. Our large models have 150-200M trainable parameters and are trained on \sim 1000h of single language pair data from multiple corpora.

Scoring For ST/SST, we evaluate detokenized case-sensitive BLEU (Post, 2018). For SST, we additionally evaluate Average Lagging (AL) (Ma et al., 2020a). For S2ST, we evaluate ASR-BLEU by transcribing the generated speech and then evaluating the BLEU of this transcription.

5.2 Results

Toolkit Comparison Table 2 summarizes ESPnet-ST-v2 performance, showing one best example model (§4) for each task. ESPnet-ST-v1,

MODEL	KD	BT	ENS	BLEU \uparrow
IWSLT'21 (Top 3 of 6)				
1 Volctrans E2E \dagger	✓	-	✓	24.3
2 OPPO Cascade \dagger	✓	✓	✓	22.6
3 Volctrans Cascade \dagger	✓	✓	✓	22.2
ESPnet-ST-v2				
A Base CA	-	-	-	23.2
B Base MCA	-	-	-	23.6
C Large CA	-	-	-	24.3
D Large MCA	-	-	-	25.1
E MBR (A+B+C+D)	-	-	✓	25.4

Table 4: Base and large CTC/attention (CA) and Multi-decoder CTC/attention (MCA) models compared to top IWSLT 2021 systems for the given segmentation tst2020 En-De test set. KD=Knowledge Distillation, BT=Back-Translation, Ens=Ensemble. \dagger Uses WMT MT data.

MODEL	BSz	BLEU \uparrow /AL \downarrow
Blockwise Attn Enc-Dec (BAED)	40	22.8 / 3.23
Label-Sync Blockwise CTC/Attn (LBCA)	40	24.4 / 3.23
Time-Sync Blockwise CTC/Attn (TBCA)	40	24.6 / 2.34
Blockwise Transducer (BT)	40	22.9 / 2.37
Blockwise Attn Enc-Dec (BAED)	20	21.0 / 2.77
Label-Sync Blockwise CTC/Attn (LBCA)	20	22.9 / 2.77
Time-Sync Blockwise CTC/Attn (TBCA)	20	22.8 / 1.63
Blockwise Transducer (BT)	20	20.9 / 1.71

Table 5: Example SST models – results on MuST-C-v2 En-De tst-COMMON. BSz=Block Size.

Fairseq, and NeurST models are also referenced for comparison. On ST/SST, ESPnet-ST-v2 is 4-7 BLEU higher with 4.5 sec lower AL.³ On S2ST ESPnet-ST-v2 is on par with Fairseq.

ST Table 3 shows a variety of approaches, of which the CTC/attention and Multi-decoder CTC/attention (MCA) models show the strongest performances. In Table 4, we scale these two approaches by training on larger corpora and increasing model capacity – *our large MCA model outperforms the best IWSLT 2021 offline track submission on the 2020 test set with given segmentation.*

SST Table 5 shows a variety of approaches, of which the blockwise Transducer (BT) and time-synchronous blockwise CTC/attention (TBCA) models have the lowest AL. We choose to scale the TBCA to compare with IWSLT submissions due to its superior translation quality, but note that the BT has lower computational overhead due pri-

³This comparison refers to the originally published results from the toolkit description papers. Note that subsequent works using these toolkits have improved the performance.

MODEL	SSL	LLM	KD	BLEU \uparrow / AL \downarrow
IWSLT'22 (Top 3 of 5)				
1 CUNI-KIT E2E	✓	✓	-	31.5 / 1.93
2 UPV Cascade \dagger	-	-	-	27.8 / 1.93
3 FBK E2E \dagger	-	-	✓	25.0 / 1.99
ESPnet-ST-v2				
A Base TBCA	-	-	-	24.7 / 1.93
B Large TBCA	-	-	-	26.6 / 1.93

Table 6: Base and large time-sync CTC/attention (TBCA) models compared to top IWSLT 2022 systems for the medium latency regime. Evaluated on En-De tst-COMMON-v2. SSL=Speech Self-Supervised Learning, LLM=Large Pre-trained Language Model, KD=Knowledge Distillation. \dagger Uses WMT MT data.

MODEL	TYPE	ASR-BLEU \uparrow
Prior Works		
1 Translatotron (Jia et al., 2019)	Spectral	14.4
2 Translatotron2 (Jia et al., 2022a)	Spectral	30.3
3 Translatotron2+ (Inaguma et al., 2022)	Spectral	32.8
4 Speech-to-Unit (Lee et al., 2022a)	Discrete	30.8
5 UnitY (Inaguma et al., 2022)	Discrete	32.3
ESPnet-ST-v2		
A Attn Enc-Dec (Translatotron)	Spectral	16.6
B Multi-Decoder (Translatotron2)	Spectral	24.3
C Attn Enc-Dec (Speech-to-Unit)	Discrete	31.3
D Multi-Decoder (UnitY)	Discrete	32.0

Table 7: Example S2ST models – results on CVSS-C Es-En test set. Prior works shown for comparison.

FRONTEND	DISCRETE UNIT	ASR-BLEU \uparrow
FBANK	HuBERT	14.8
wav2vec2 \dagger	HuBERT	21.2
HuBERT \dagger	HuBERT	21.4
mHuBERT	HuBERT	21.5
WavLM \dagger	HuBERT	22.8
FBANK	WavLM	15.0
wav2vec2 \dagger	WavLM	21.6
HuBERT \dagger	WavLM	22.1
mHuBERT	WavLM	22.0
WavLM \dagger	WavLM	23.1

Table 8: Ablation on different types of SSL for the frontend and discrete unit portions of S2ST models. \dagger Trained with large settings, others with base settings.

marily to the lack of source-target computation; AL is non-computation aware. In Table 6, we fit the TBCA to the 2 second AL latency regime by selecting a blocksize of 32 and scale it with more data and model capacity – *our large TBCA model would have ranked 3rd out of 6 amongst IWSLT 2022 submissions without using any SSL / LLM representations or knowledge distillation.*

S2ST Table 7 shows a variety of approaches compared to prior works with comparable architectures – *our S2ST models are generally on par with prior works which are considered state-of-the-art*. In fact, all of our models slightly outperform their respective prior works except for Translatotron 2. Further, in Table 8 we ablate a range of SSL types for both the frontend and discrete units demonstrating the flexibility of our toolkit.

6 Conclusion

We presented ESPnet-ST-v2 which now supports offline speech translation, simultaneous speech translation, and offline speech-to-speech translation. ESPnet-ST-v2 will continue to grow to support the community’s interests. Future updates may include more new tasks, such as simultaneous speech-to-speech translation, and cross-toolkit integrations via TorchAudio.

Limitations

The first set of limitations to be aware of are data-related. Although prior works have shown the feasibility of building E2E systems without source language transcriptions (Lee et al., 2022b; Chen et al., 2022; Zhang et al., 2021), in this work we only investigate cases where triplet data (source speech, source transcript, target translation) is available for ST/SST and where quadruplet data (source speech, source transcript, target translation, target speech) is available for S2ST.

The second set of limitations to be aware of are evaluation-related. For SST, we follow prior works (Ma et al., 2020a; Wang et al., 2020; Anastasopoulos et al., 2022) and evaluate AL which is a measure of how much the system outputs lags behind the amount of input read. Notably, this does not consider the actual computation time and only the input-to-output ratio. For S2ST, we follow prior works (Jia et al., 2022a; Inaguma et al., 2022) and evaluate ASR-BLEU. This evaluation is dependent on an ASR system, which is not standardized across prior works. And further, our evaluation of S2ST outputs does not include naturalness. Finally, in this work we have not conducted any human evaluation of translation outputs.

Acknowledgements

Brian Yan and Shinji Watanabe are supported by the Human Language Technology Center of Excellence. This work also used the Extreme

Science and Engineering Discovery Environment (XSEDE) (Townes et al., 2014), which is supported by National Science Foundation grant number ACI-1548562; specifically, the Bridges system (Nystrom et al., 2015), as part of project cis210027p, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center. This work also used GPUs donated by the NVIDIA Corporation.

References

- Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A Smith, and David Yarowsky. 1999. *Statistical machine translation*. In *Final Report, JHU Summer Workshop*, volume 30.
- Antonios Anastasopoulos, Loïc Barrault, Luisa Bentivogli, Marceley Zanon Boito, Ondřej Bojar, Roldano Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, Clara Emmanuel, Yannick Estève, Marcello Federico, Christian Federmann, Souhir Gahbiche, Hongyu Gong, Roman Grundkiewicz, Barry Haddow, Benjamin Hsu, Dávid Javorský, Věra Kloudová, Surafel Lakew, Xutai Ma, Prashant Mathur, Paul McNamee, Kenton Murray, Maria Nădejde, Satoshi Nakamura, Matteo Negri, Jan Niehues, Xing Niu, John Ortega, Juan Pino, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Yogesh Virkar, Alexander Waibel, Changhan Wang, and Shinji Watanabe. 2022. *Findings of the IWSLT 2022 evaluation campaign*. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 98–157, Dublin, Ireland (in-person and online). Association for Computational Linguistics.
- Luisa Bentivogli, Mauro Cettolo, Marco Gaido, Alina Karakanta, Alberto Martinelli, Matteo Negri, and Marco Turchi. 2021. Cascade versus direct speech translation: Do the differences still make a difference? 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 2873–2887.
- Dan Berrebbi, Jiatong Shi, Brian Yan, Osbel López-Francisco, Jonathan Amith, and Shinji Watanabe. 2022. *Combining Spectral and Self-Supervised Features for Low Resource Speech Recognition and Translation*. In *Proc. Interspeech 2022*, pages 3533–3537.
- Alan W Black, Ralf D Brown, Robert Frederking, Rita Singh, John Moody, and Eric Steinbrecher. 2002. Tongues: Rapid development of a speech-to-speech translation system. In *Proceedings of Second International Conference on Human Language Technology Research HLT*, pages 183–186.

- Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2021. Direct simultaneous speech-to-text translation assisted by synchronized streaming asr. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4618–4624.
- Peng-Jen Chen, Kevin Tran, Yilin Yang, Jingfei Du, Justine Kao, Yu-An Chung, Paden Tomasello, Paul-Ambroise Duquenne, Holger Schwenk, Hongyu Gong, et al. 2022. Speech-to-speech translation for a real-world unwritten language. *arXiv preprint arXiv:2211.06474*.
- Siddharth Dalmia, Brian Yan, Vikas Raunak, Florian Metze, and Shinji Watanabe. 2021. [Searchable hidden intermediates for end-to-end models of decomposable sequence tasks](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1882–1896, Online. Association for Computational Linguistics.
- Keqi Deng, Shinji Watanabe, Jiatong Shi, and Siddhant Arora. 2022. [Blockwise Streaming Transformer for Spoken Language Understanding and Simultaneous Speech Translation](#). In *Proc. Interspeech 2022*, pages 1746–1750.
- Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. [MuST-C: a Multilingual Speech Translation Corpus](#). 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 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.
- Patrick Fernandes, António Farinhas, Ricardo Rei, José De Souza, Perez Ogayo, Graham Neubig, and André FT Martins. 2022. Quality-aware decoding for neural machine translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412.
- Marco Gaido, Mauro Cettolo, Matteo Negri, and Marco Turchi. 2021. [CTC-based compression for direct speech translation](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 690–696, Online. Association for Computational Linguistics.
- Alex Graves. 2012. Sequence transduction with recurrent neural networks. *ArXiv*, abs/1211.3711.
- 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*, pages 5036–5040.
- Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, et al. 2021. Recent developments on espnet toolkit boosted by conformer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878. IEEE.
- Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan. 2020. ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 7654–7658. IEEE.
- Tomoki Hayashi, Ryuichi Yamamoto, Takenori Yoshimura, Peter Wu, Jiatong Shi, Takaaki Saeki, Yooncheol Ju, Yusuke Yasuda, Shinnosuke Takamichi, and Shinji Watanabe. 2021. ESPnet2-TTS: Extending the edge of TTS research. *arXiv preprint arXiv:2110.07840*.
- Xuedong Huang, James Baker, and Raj Reddy. 2014. [A historical perspective of speech recognition](#). *Communications of the ACM*, 57(1):94–103.
- Hirofumi Inaguma, Tatsuya Kawahara, and Shinji Watanabe. 2021a. Source and target bidirectional knowledge distillation for end-to-end speech translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1872–1881.
- Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplín, Tomoki Hayashi, and Shinji Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. *arXiv preprint arXiv:2004.10234*.
- Hirofumi Inaguma, Sravya Popuri, Iliia Kulikov, Peng-Jen Chen, Changhan Wang, Yu-An Chung, Yun Tang, Ann Lee, Shinji Watanabe, and Juan Pino. 2022. UnitY: Two-pass direct speech-to-speech translation with discrete units. *arXiv preprint arXiv:2212.08055*.
- Hirofumi Inaguma, Brian Yan, Siddharth Dalmia, Pengcheng Guo, Jiatong Shi, Kevin Duh, and Shinji Watanabe. 2021b. [ESPnet-ST IWSLT 2021 offline speech translation system](#). In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 100–109, Bangkok, Thailand (online). Association for Computational Linguistics.
- Javier Iranzo-Sánchez, Javier Jorge, Pau Baquero-Arnal, Joan Albert Silvestre-Cerdà, Adrià Giménez, Jorge Civera, Albert Sanchis, and Alfons Juan. 2021. Streaming cascade-based speech translation leveraged by a direct segmentation model. *Neural Networks*, 142:303–315.

- Ye Jia, Michelle Tadmor Ramanovich, Tal Remez, and Roi Pomerantz. 2022a. Translatotron 2: High-quality direct speech-to-speech translation with voice preservation. In *International Conference on Machine Learning*, pages 10120–10134. PMLR.
- Ye Jia, Michelle Tadmor Ramanovich, Quan Wang, and Heiga Zen. 2022b. CVSS corpus and massively multilingual speech-to-speech translation. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pages 6691–6703.
- Ye Jia, Ron J Weiss, Fadi Biadisy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. 2019. Direct speech-to-speech translation with a sequence-to-sequence model. *Proc. Interspeech 2019*, pages 1123–1127.
- Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyang Jiang, Masao Someki, Nelson Enrique Yalta Soplín, Ryuichi Yamamoto, Xiaofei Wang, et al. 2019. A comparative study on transformer vs RNN in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456. IEEE.
- Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J Han, and Shinji Watanabe. 2023. E-branchformer: Branchformer with enhanced merging for speech recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 84–91. IEEE.
- Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033.
- 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 preprint arXiv:1909.09577*.
- Ann Lee, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Sravya Popuri, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, et al. 2022a. Direct speech-to-speech translation with discrete units. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3327–3339.
- Ann Lee, Hongyu Gong, Paul-Ambroise Duquenne, Holger Schwenk, Peng-Jen Chen, Changhan Wang, Sravya Popuri, Yossi Adi, Juan Pino, Jiatao Gu, and Wei-Ning Hsu. 2022b. [Textless speech-to-speech translation on real data](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 860–872, Seattle, United States. Association for Computational Linguistics.
- Jaesong Lee and Shinji Watanabe. 2021. Intermediate loss regularization for CTC-based speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE.
- Danni Liu, Gerasimos Spanakis, and Jan Niehues. 2020a. [Low-Latency Sequence-to-Sequence Speech Recognition and Translation by Partial Hypothesis Selection](#). In *Proc. Interspeech 2020*, pages 3620–3624.
- Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.
- Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. Simuleval: An evaluation toolkit for simultaneous translation. In *Proceedings of the EMNLP*.
- Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. SimulMT to simulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. 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 582–587.
- Jan Niehues, Rolando Cattoni, Sebastian Stüker, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. [The IWSLT 2018 evaluation campaign](#). In *Proceedings of the 15th International Conference on Spoken Language Translation*, pages 2–6, Brussels. International Conference on Spoken Language Translation.
- Nicholas A Nystrom, Michael J Levine, Ralph Z Roskies, and J Ray Scott. 2015. [Bridges: a uniquely flexible hpc resource for new communities and data analytics](#). In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, pages 1–8.
- Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.
- Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. 2022. Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding. In *International Conference on Machine Learning*, pages 17627–17643. PMLR.
- Adam Polyak, Yossi Adi, Jade Copet, Eugene Kharonov, Kushal Lakhotia, Wei-Ning Hsu, Abdelrahman Mohamed, and Emmanuel Dupoux. 2021. Speech Resynthesis from Discrete Disentangled Self-Supervised Representations. In *Proc. Interspeech 2021*.

- 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.
- Ramon Sanabria and Florian Metze. 2018. Hierarchical multitask learning with CTC. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 485–490. IEEE.
- George Saon, Zoltán Tüske, and Kartik Audhkhasi. 2020. [Alignment-length synchronous decoding for RNN transducer](#). In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7804–7808.
- Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjali Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.
- J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. 2014. [Xsede: Accelerating scientific discovery](#). *Computing in Science & Engineering*, 16(5):62–74.
- Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. 2021. Streaming transformer ASR with blockwise synchronous beam search. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 22–29. IEEE.
- 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 Waibel, Ajay N Jain, Arthur E McNair, Hiroaki Saito, Alexander G Hauptmann, and Joe Tebelskis. 1991. [JANUS: a speech-to-speech translation system using connectionist and symbolic processing strategies](#). In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 793–796 vol.2.
- Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. 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: System Demonstrations*, pages 33–39.
- Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. 2017. Tacotron: Towards end-to-end speech synthesis. *Proc. Interspeech 2017*, pages 4006–4010.
- Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. 2017. [Hybrid CTC/attention architecture for end-to-end speech recognition](#). *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.
- 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.
- Jian Xue, Peidong Wang, Jinyu Li, Matt Post, and Yashesh Gaur. 2022. [Large-Scale Streaming End-to-End Speech Translation with Neural Transducers](#). In *Proc. Interspeech 2022*, pages 3263–3267.
- Brian Yan, Siddharth Dalmia, Yosuke Higuchi, Graham Neubig, Florian Metze, Alan W Black, and Shinji Watanabe. 2023. CTC alignments improve autoregressive translation. *The 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.
- Brian Yan, Patrick Fernandes, Siddharth Dalmia, Jia-tong Shi, Yifan Peng, Dan Berrebbi, Xinyi Wang, Graham Neubig, and Shinji Watanabe. 2022. CMU’s iwslt 2022 dialect speech translation system. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 298–307.
- Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung yi Lee. 2021a. SUPERB: Speech processing universal performance benchmark. *Proc. Interspeech 2021*.
- Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhersch, David Pollack, Dmitry Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. 2021b. Torchaudio: Building blocks for audio and speech processing. *arXiv preprint arXiv:2110.15018*.
- Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller, Ngoc-Quan Pham, Sebastian Stüker, and Alex Waibel. 2018. Open source toolkit for speech to text translation. *Prague Bull. Math. Linguistics*, 111:125–135.

Chen Zhang, Xu Tan, Yi Ren, Tao Qin, Kejun Zhang, and Tie-Yan Liu. 2021. UWSpeech: Speech to speech translation for unwritten languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14319–14327.

Weitai Zhang, Zhongyi Ye, Haitao Tang, Xiaoxi Li, Xinyuan Zhou, Jing Yang, Jianwei Cui, Pan Deng, Mohan Shi, Yifan Song, et al. 2022. The USTC-NELSLIP offline speech translation systems for IWSLT 2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 198–207.

Chengqi Zhao, Mingxuan Wang, Qianqian Dong, Rong Ye, and Lei Li. 2021. NeurST: Neural speech translation toolkit. In *the 59th Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*.

A Appendix

A.1 Reproducibility

Table 9 shows the hyperparameters for the models presented in §5. All of our data preparation scripts are available in ESPnet: <https://github.com/espnet/espnet/tree/master/egs2>.

Model	Task	Encoder(s)	Decoder(s)	Frontend	Pre-Train Init	Multi-Obj	Src BPE	Tgt BPE	# Params
AED (Table 3)	ST	12 lyr, 4 head, 256 adim	(ASR) 6 lyr, 4 head (ST) 6 lyr, 4 head	FBANK	ASR Enc/Dec	ASR	4k	4k	60M
MAED (Table 3)	ST	(ASR) 12 lyr, 4 head, 256 adim (MT) 2 lyr, 4 head, 256 adim	(ASR) 6 lyr, 4 head (MT) 6 lyr, 4 head	FBANK	ASR Enc/Dec	ASR	4k	4k	60M
CA (Table 3)	ST	18 lyr, 4 head, 256 adim	(ASR) 6 lyr, 4 head (ST) 6 lyr, 4 head	FBANK	ASR Enc/Dec	ASR	4k	4k	70M
MCA (Table 3)	ST	(ASR) 18 lyr, 4 head, 256 adim (MT) 4 lyr, 4 head, 256 adim	(ASR) 6 lyr, 4 head (MT) 6 lyr, 4 head	FBANK	ASR Enc/Dec/CTC	ASR	4k	4k	70M
T (Table 3)	ST	18 lyr, 4 head, 256 adim	1 lyr, 512 dim, 640 joint (ASR) 6 lyr, 4 head	FBANK	ASR Enc/Dec/CTC	ASR	4k	4k	70M
Large CA (Table 4)	ST	18 lyr, 8 head, 512 adim	(ASR) 6 lyr, 4 head (ST) 6 lyr, 4 head	HuBERT	ASR Enc/Dec/CTC	ASR	8k	16k	210M
Large MCA (Table 4)	ST	(ASR) 18 lyr, 8 head, 512 adim (MT) 4 lyr, 8 head, 512 adim	(ASR) 6 lyr, 8 head (MT) 6 lyr, 8 head	HuBERT	ASR Enc/Dec/CTC	ASR	8k	8k	210M
BAED (Table 5)	SST	18 lyr, 4 head, 256 adim	6 lyr, 4 head	FBANK	ASR Enc lyr 1-12	-	4k	4k	70M
LBCA (Table 5)	SST	18 lyr, 4 head, 256 adim	6 lyr, 4 head	FBANK	ASR Enc lyr 1-12	-	4k	4k	70M
TBCA (Table 5)	SST	18 lyr, 4 head, 256 adim	6 lyr, 4 head	FBANK	ASR Enc lyr 1-12	-	4k	4k	70M
BT (Table 5)	SST	18 lyr, 4 head, 256 adim	1 lyr, 4 head, 640 joint	FBANK	ASR Enc lyr 1-12	-	4k	4k	40M
Large TBCA (Table 6)	SST	18 lyr, 8 head, 512 adim	6 lyr, 8 head	FBANK	ASR Enc lyr 1-12	-	8k	8k	150M
Translatotron (Table 7)	S2ST	12 lyr, 4 head, 256 adim	6 lyr, 1024 dim	FBANK	-	ASR, ST	7k	500	80M
Translatotron2 (Table 7)	S2ST	16 lyr, 4 head, 256 adim	(ST) 6 lyr, 4 head (TTS) 2 lyr, 1024 dim	FBANK	-	ASR, ST	7k	500	50M
Speech-to-Unit (Table 7)	S2ST	12 lyr, 4 head, 512 adim	6 lyr, 8 head	FBANK	-	ASR, ST	7k	500	40M
UnitY (Table 7)	S2ST	(ST) 16 lyr, 4 head, 256 adim (T2U) 2 lyr, 4 head, 256 adim	(ST) 4 lyr, 4 head (T2U) 2 lyr, 8 head	FBANK	-	ASR, ST	7k	500	40M

Table 9: ST, SST, and S2ST model hyperparameters. Parameter counts are rounded to the nearest 10 million.

CB2: Collaborative Natural Language Interaction Research Platform

Jacob Sharf, Mustafa Omer Gul, and Yoav Artzi

Department of Computer Science and Cornell Tech, Cornell University
jacobsharf@gmail.com {momergul, yoav}@cs.cornell.edu

Abstract

CB2 is a multi-agent platform to study collaborative natural language interaction in a grounded task-oriented scenario. It includes a 3D game environment, a backend server designed to serve trained models to human agents, and various tools and processes to enable scalable studies. We deploy CB2 at <https://cb2.ai> as a system demonstration with a learned instruction following model.

1 Introduction

Collaborative grounded natural language interactions involve multiple agents, either human or machine, working together to complete tasks while coordinating using natural language. A key obstacle in studying such scenarios is building the research interaction platform, a significant design and engineering undertaking. This requires building and designing the interaction environment, the task the agents collaborate on, an interface for both machine learning models and human agents, and a process to onboard human agents. Each aspect dramatically influences the interaction and language elicited, and is critical to get right.

We introduce CB2, a platform for the study of collaborative grounded natural language interaction, and demonstrate its use through the deployment of a learned collaborative natural language agent. CB2 largely instantiates the CEREALBAR scenario (Suhr et al., 2019),¹ but is implemented from scratch to emphasize research accessibility. CB2 is a customizable, scalable, and complete research platform, including server and clients for multi-agent human-machine interactions, tools for real-time data management, and processes to onboard crowdsourcing workers.

The CB2 scenario poses learning and reasoning challenges, as well as opportunities. Comprehending and producing instructions in CB2 requires

¹CB2 introduces several optional modifications to CEREALBAR aimed at richer language and tighter collaboration.

addressing the symbol grounding problem (Harnad, 1990), which is studied extensively in the instruction following (e.g., Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Misra et al., 2017; Fried et al., 2018) and generation (e.g., Mei et al., 2016; Wang et al., 2021) literature. However, the collaborative scenario remains relatively understudied. Collaboration is not simply an added complication, but dramatically alters both interaction and learning through joint presence and action. It allows the instructor to ad-hoc modify the tasks they delegate based on the follower behavior, potentially recovering from system failures. At the same time, this adaptation creates constant distribution shift, a significant generalization challenge. Learning is also drastically transformed through collaboration. The constant engagement of other agents (including humans), the ability to modify delegation strategies, and the shared task-based incentives bring about within-interaction signals that can be used for continual learning, reducing the dependency on annotated data and enabling model adaptation.

We deploy a demonstration of CB2 with a learned baseline instruction following agent (Section 7). Players can connect to CB2 and collaborate with our agent or other human agents at <https://cb2.ai/>.² The CB2 platform is available at <https://github.com/lil-lab/cb2>. A video demonstration of CB2 is available at https://youtu.be/tALpX_KKmIw.

2 Related Work

CB2 is a re-implementation and extension of CEREALBAR, a scalable platform to study natural language instruction collaboration (Suhr et al., 2019). CEREALBAR was used to study instruction following (Suhr et al., 2019; Suhr and Artzi, 2022), instruction generation (Kojima et al., 2021), and linguistic change (Effenberger et al., 2021).

²Our deployment has received IRB exemption. All recorded data is anonymized.

CB2 is related to instruction following environments, such as SAIL (MacMahon et al., 2006), R2R (Anderson et al., 2018), RxR (Ku et al., 2020), and ALFRED (Shridhar et al., 2020). In contrast, CB2 is focused on embodied multi-agent collaborations, including with human agents.

Symbol grounding (Harnad, 1990), a core challenge in CB2, was studied extensively in the single-agent context of instruction following (e.g., Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Fried et al., 2018; Blukis et al., 2018) and generation (e.g., Daniele et al., 2016; Kojima et al., 2021; Wang et al., 2021). The CB2 scenario emphasizes multi-agent collaboration, an aspect that is significantly less studied with natural language instruction. The Cards corpus (Djalali et al., 2012; Potts, 2012) presents a related scenario, which has been used for linguistic analysis. A related problem is studied by the emergent communication literature (Lazaridou et al., 2017; Andreas et al., 2017; Lazaridou and Baroni, 2020), but with less focus on collaboration with human agents. Natural language collaboration between agents with asymmetric capabilities has also been studied with Minecraft-based scenarios (Narayan-Chen et al., 2019; Jayannavar et al., 2020; Kiseleva et al., 2022). CB2 differs from these in allowing both agents to effect changes on the environment, enabling ad-hoc modification and delegation of tasks.

3 Interaction Scenario

CB2 largely implements the interaction scenario introduced by (Suhr et al., 2019) in the CEREALBAR environment with several modifications. The interaction takes place in a procedurally generated spatial environment and includes two agents that collaborate together to complete card collection tasks and coordinate using natural language. Figure 1a shows an instance of the environment.

The environment is a procedurally generated 3D map made of a grid of hexagons (Figure 1a). It includes lakes, mountains (Figure 1c), paths, open spaces, and landmarks. A new environment is generated for each game. CB2 includes improved visuals and generation compared to CEREALBAR. For example, CB2 map generation includes semantic biases: houses are generated to cluster together and form towns (Figure 1b) and paths are generated to connect between meaningful areas in the map, such as towns and mountains. Landmark instances vary visually to elicit richer language. For example,

houses are generated with different roof colors and number of floors (Figure 1b). The environment also includes randomly placed cards (Figure 1d). Each card shows 1–3 copies of one of a few possible shapes in one of a few possible colors.

The interaction involves two agents, a leader (Figure 1f) and a follower (Figure 1g), that collaborate together to complete tasks, but differ in their observations of the environments and abilities. Both the leader and the follower move in the environment, by moving between neighboring hexagons or by turning in place to change orientation. The agents select and deselect cards by moving over them (Figure 1e).

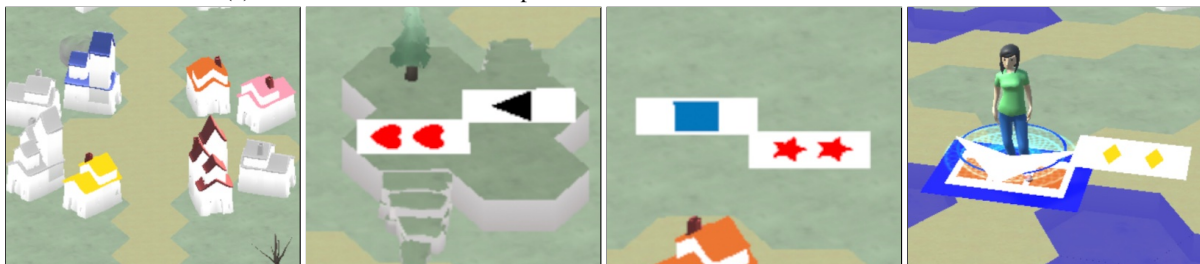
The goal of the agents is to select valid sets of cards. A valid set includes three cards, where each color, shape, and count are unique (Figure 1i). The agents select sets together. When the currently selected cards form a valid set, they disappear, the agents together receive one point, three new randomly selected cards appear in random positions, and the agents receive additional turns. The number of turns added diminishes with each set completion. Asymmetries between the two agents make collaboration critical for success.

The leader sees a complete overhead view of the environment (Figure 1a), while the follower only sees what is ahead from a first-person view (Figure 1h). CB2 introduces two optional observability features not present in CEREALBAR. First, the patterns on unselected cards may be hidden from the follower, instead displaying a question mark on all cards. Second, CB2 allows to control how far the follower sees ahead of them with a fog that is present only in the follower view. The observability gap means the leader is in charge of planning how the agents operate. If the follower acts independently of the leader plans, the interaction will be suboptimal, because follower actions are likely to conflict with leader actions and the partial view of the environment does not allow for optimal planning of goals and movement.

The agents move in turns, with a limited number of steps per turn. Each movement (forward, left, right, or backward) consumes a single step. Turns are time limited to keep the interaction moving and avoid long wait periods for the inactive agent. The exact time budget is customizable, but we generally provide significantly more time for the leader turns, so they can plan as needed. Turns alternate between the follower and leader. The follower has



(a) An overhead view of a complete environment with the leader user interface.



(b) A cluster of houses.

(c) A mountain with ramps.

(d) Cards in the environment.

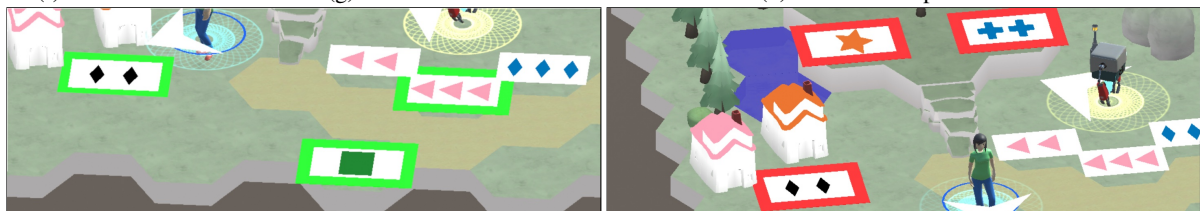
(e) The leader selecting a card.



(f) The leader character.

(g) The follower character.

(h) The follower point of view.



(i) Valid (left) and invalid (right) sets of selected cards.

Figure 1: Images of the game environment and UI. All images are taken from the same environment state.

significantly more steps than the leader per turn. This means the follower is able to move further in each turn, and potentially accomplish much more

in each turn. This ability gap makes it critical for the leader to collaborate with the follower, rather than ignore the follower and attempt to accomplish

tasks on their own, a suboptimal strategy.

The agents coordinate via uni-directional natural language instruction, the only form of coordination available. During a leader turn, in addition to moving in the environment, the leader can send text instructions to the follower. The follower executes the leader instructions and indicates when an instruction is complete. The leader can queue multiple instructions, but the follower only sees past instructions and the one they are currently executing. Because the follower does not see future instructions, alignment between the actions recorded and the instruction displayed is guaranteed. The leader can also cancel the instruction the follower is executing alongside all future instructions in the queue during the follower turn. This is intended to halt very bad executions, and reduce their overall cost, for example by having to correct drastic departures from the leader plan.

Instruction writing and sending by the leader, and marking them as complete by the follower do not consume steps. Leaders may write as many instructions as they wish during a single turn, and followers are not taxed if the tasks are given in multiple instructions that they need to mark as complete. Exempting the language channel from the budget of actions per turn aims to reduce the influence of the turn systems on the language produced. The combination of collaboration incentives (i.e., because of the capability differences between the agents) and the exclusivity of the language channel for communication makes effective natural language instruction essential for successful interactions.³

4 Framework Implementation

The CB2 framework has three main components: a Python server, a Unity client, and a Python headless client. The game logic is orchestrated from the server, allowing to customize the interaction without modifying Unity code. The Python client simplifies the interaction between learning processes and the system, for example during reinforcement learning. [Figure 2](#) visualizes the architecture.

CB2’s design emphasizes customizability, as much as possible, without modifying Unity code, a skill that is less common among researchers. This motivates placing the game logic on the Python

³Depending on the environment configuration, it is possible for one of the agents to operate alone if the cards forming a set are really close and the other agents does not move. This can allow 1–2 set completions. A higher score without collaboration via language coordination is extremely unlikely.

server, a decision that dictates the client-server communication design. However, modifications that require updating the client user interface, such as adding bi-directional communication or translating the UI to other languages, do require modifying Unity coding.

4.1 Server

The server architecture is split into modules by logical function. We use asynchronous coroutines to reduce latency efficiently and keep the compute needs small. The platform is parameterized via a configuration file that is loaded by the server.

Map Generation Map generation is relatively expensive compared to other processes on the server, mainly because we may use multiple search iterations for routing paths between landmarks and to prevent the leader or follower from spawning in closed-off regions. We mitigate potential lag because of server load by preparing a pool of maps in advance, which we refill during idle periods.

Player Lobbies The server supports multiple lobbies concurrently. Separate lobbies provide different player pairing strategies, such as for human-human and human-model games. Players wait in a lobby until they are paired for a game. Each lobby maintains multiple queues for pairing players and assigning roles according to their experience or other information. For example, by default, we distinguish between expert and novice players, and prioritize pairing experts as leaders with novices as followers. Each lobby maintains active game rooms of different types, such as for standard games, tutorials, game replays, and custom scenarios. Each game room contains a game state machine and websocket connections to the clients.

Data Storage Game events are recorded into an sqlite3 database, which allows for efficient interaction with game data. Each game is represented as a linear list of events, which can be replayed to recreate game state at a particular moment in time.

Data Portal The data portal provides an interface to view game records and statistics. The web data browser shows game-specific recordings, including turns, instructions, and individual player actions. Each game record also includes a link to launch a game replay using the web client. There is also a web page with live statistics, such as the mean and median scores, and a page to download an archive of all server data. The data portal also provides an

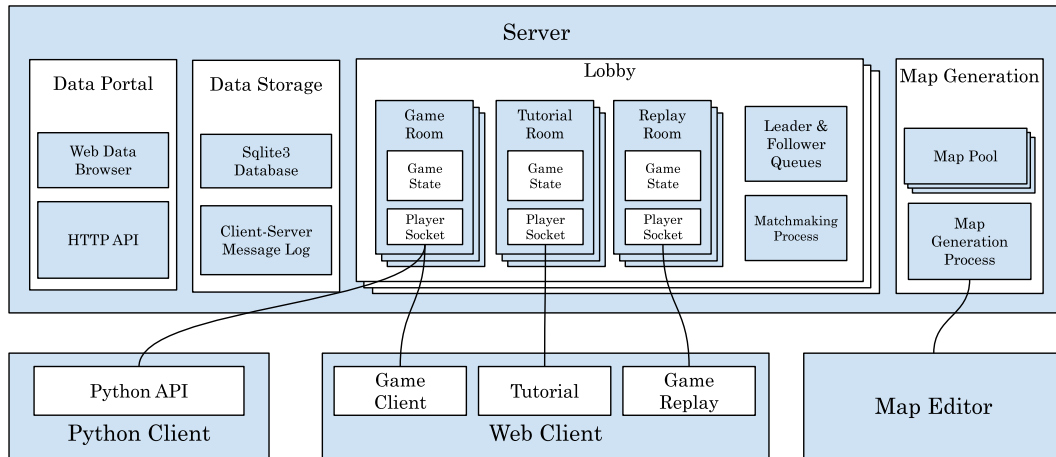


Figure 2: The CB2 system architecture.

HTTP API for programmatic data access.

Map and Scenario Editor Maps are generated procedurally by default. CB2 also provides a map editor for researchers to place users in controlled scenarios. A real-time API allows attaching to an interaction and update the map in response to the game state, enabling dynamic interactions.

4.2 Web Client

The web client is developed using the Unity game engine, and is packaged as a WebAssembly binary. The client receives game states, actions, and instructions from the server. We design the client to be thin, putting as much of the game logic and configuration on the server as possible, so that changes to game logic can be made purely in Python. We designed the gameplay user interface (UI) to be accessible and easy to learn by incorporating feedback from players. All UI elements are clustered together, and have keyboard shortcuts. Figure 1a shows the leader interface during a leader turn.

Beyond gameplay, the web client provides a tutorial to onboard players to the game by stepping them through a game interaction accompanied by prompts and tooltips. The tutorial flow is specified on the server, and can be modified easily. For example, rephrasing the tutorial instructions or translating them to other languages is relatively simple and can be achieved by updating the specifications in the server code. The web client also provides game replay, which is activated by adding URL parameters when the HTML page is loaded. The parameters are added automatically to links in the web data browser (Section 4.1).

```
def PlayGameAsFollower(game):
    game_state = game.initial_state()
    # The game starts with the leader's turn.
    # Wait for follower's turn by executing a noop.
    game_state = game.step(Action.NoopAction())
    while not game.over():
        action = get_action(game_state)
        game_state = game.step(action)
    (_, _, turn_state, _, _) = game_state
    print(f"Game over. Score: {turn_state.score}")
```

Figure 3: Example code using the Python API.

4.3 Python Client

The programmatic Python client API supports fast lightweight interaction with the game. It is designed for machine learning processes that require interacting with the game environment, such as reinforcement learning (Sutton and Barto, 1998), and can be used to deploy agents interacting with human players or agent-agent interactions. Interaction through this API are similar to interactions with the Unity client, except that recording is optional to reduce overhead. When recorded, they can be replayed using the Unity client. We also provide an OpenAI Gym-style wrapper for the Python API. Figure 3 shows example code.

5 Example Task Formulations

CB2 is well suited to study a variety of tasks, with emphasis on learning and evaluation in collaborative interactions with human agents, such as:

Instruction Following The task of instruction following is to map a start state observation from the follower perspective and a leader instruction to a sequence of actions. After each action, the agent receives a new observation. Suhr et al. (2019)

studied this problem with CEREALBAR by learning from recorded human-human interactions, and [Suhr and Artzi \(2022\)](#) studied it within a continual learning from human feedback scenario. Both approaches were evaluated by deploying follower agents to interact with human leaders.

Instruction Generation The task of instruction generation is to generate a leader instruction for the follower to execute given an observation of the world state from the leader perspective. This requires planning the cards the two agents should select, divide the tasks, plan trajectories, and express the intended follower trajectory in a natural language instruction. [Kojima et al. \(2021\)](#) focused on the problem of mapping deterministically generated plans to natural language instructions, and proposed a continual learning approach for learning by observing human follower behavior.

Emergent Communication CB2 is particularly well suited to study emergent communication in multi-agent systems ([Lazaridou and Baroni, 2020](#)). The goal is to jointly learn separate models for the leader and follower. The two models generate actions to move in the world. The leader model additionally generates instructions, which the follower model is conditioned on. The learning can be driven by performance in the game. CB2 easily allows to integrate human agents into the learning and evaluation processes, bringing natural human language into the process. Alternating between interaction between agent-agent and agent-human interactions has the potential to address the language drift problem ([Lee et al., 2019](#)).

6 Crowdsourcing Process

CB2 poses several relatively demanding crowdsourcing tasks. Human-human interactions require pairing two workers for real-time play over extended time. We design a process to collect CB2 interactions via crowdsourcing, either for games where both roles are controlled by human players, or where one of the sides is controlled by a learned model. The task-focused design of CB2 naturally allows an effective incentive structure by tying game performance with compensation.

The key to our process is gradual training of workers. A new worker first starts with a tutorial and a qualifier quiz that covers the relatively simple role of the follower. The follower role requires following the leader instructions by controlling the character in the game. The worker is then qualified

to the follower role only, and is paired by joining a dedicated follower-only queue in the lobby. Focusing on the follower role only simplifies the learning curve, and much of the learning required for the leader role takes place on the job, as the worker collaborates with more experienced leaders.

Once the worker displays sufficient level of performance for several games, they are invited to qualify as a leader by taking a leader tutorial and a quiz. The second tutorial is both longer and more complex than the follower tutorial, and includes both planning and instruction writing. Once the worker completes the tutorial and passes the quiz, they are qualified to the leader role, and can then participate in tasks as both leader or follower.

We design the lobby to pair workers based on experience. Because the leader role is significantly more critical to the effectiveness of the interaction and the quality of language data, we prioritize workers with better performance for it. We measure worker performance, keeping track of the mean game score in the most recent games. If two leader-qualified players are waiting in the lobby for matching, we will assign the leader role to the higher performing of the two.

The pay structure includes a base pay for connecting to the game, and an increasing bonus for each point. Both workers, the leader and follower, get the base pay and the additional bonus per point, tightly connecting compensation to their collaborative performance. Because the leader role is more complex, we provide an additional relative bonus to the worker in the leader role.

7 CB2 Demonstration Deployment

We demonstrate the functionality and potential of CB2 via deployment, including collecting a corpus of human-human interaction that we release, training a follower baseline model, and evaluating it in interaction with human leaders.

Human Games Data We follow the crowdsourcing process outlined in Section 6 to collect games between human leaders and followers. We collect 185 games containing 3,439 instructions. [Table 1](#) provides data statistics.

Model and Learning We train an instruction following model with a behavior cloning objective using the collected human-human data. We filter out poor games to improve training data quality, applying heuristics such as removing games where over 20% of instructions are cancelled. Our

Dataset	# Games	# Instructions	Mean Score	Vocabulary	Mean Instruction Length
Training Data	185	3,439	6.42 \pm 4.88	714	10.95 \pm 5.29
Human-Human Deployment	187	3,404	6.69 \pm 4.51	728	11.73 \pm 6.09
Human-Model Deployment	188	2,869	3.15 \pm 3.29	542	9.62 \pm 5.28

Table 1: Data and interaction statistics for the human-human training data, and the two side-by-side deployments.

model architecture is based on the Decision Transformer (Chen et al., 2021; Putterman et al., 2022). Follower observations are embedded using HEXACONV (Hoogeboom et al., 2018) because of the hexagonal structure of the map. The observations are centered on the follower’s position and rotated such that the follower is always facing the same direction. This baseline model conditions only on the current instruction for simplicity, similar to the model in Suhr et al. (2019). In contrast though, it does not assume full observability.

Results We deploy our baseline model as a system demonstration on Amazon Mechanical Turk. We evaluate it with 188 human-model interactions, conducted side-by-side in a randomized experiment with 187 human-human interactions. Human leaders are told that they can be matched with a human or a bot follower in the task description, but are not made aware of who they are interacting with in a specific interaction. Table 1 shows data and interaction statistics for our training data and final deployments. Overall, our models enable effective human-model collaboration in CB2, but at significantly lower performance than observed in human-human games. This is similar to the results of Suhr et al. (2019), although the numbers are not comparable because of the different environment.

Human leaders were able to infer relatively consistently the type of their partner in each interaction. This is indicated by differences in the human leader behavior when comparing human-human and human-model interactions. For instance, the vocabulary human leaders use in interactions with the model is smaller compared to when interacting with human followers and the instructions are shorter. Qualitatively, we observe that instructions in human-human interactions more often use exclamations (e.g., “oh,” “shoot,” and “oops”) and informal speech, with abbreviations such as “btw” and “lol” or words such as “chill” and “kay.” We also found that human leaders in human-human games tend to praise their partners, with words such as “awesome,” “wonderful,” “perfect” or “great” appearing uniquely in instructions from human-human games. The difference is also seen in game

statistics. For instance, 16.54% and 12.70% of the times followers and leaders selected a card in human-model games, it was to deselect an already selected card, compared to 8.68% and 8.78% for human-human games. Our results illustrate the challenge posed by CB2, and the importance of the kind of deployment CB2 enables.

8 Conclusion

CB2 is a multi-agent research platform to study natural language instruction in collaborative, embodied environments. A core objective of CB2 is to enable scalable studies where human agents interact with learned models, potentially over long periods of time. CB2 is designed to be easy to use and customize, with emphasis on accessibility for researchers with limited game development experience. It is designed from the ground up for machine learning, and includes a headless fast Python client API to support learning processes and to deploy learned models to interact with human users.

Acknowledgements

This research was supported by NSF under grant No. 1750499, ARO W911NF21-1-0106. We thank Alane Suhr and Noriyuki Kojima for technical discussions and utility code, and the participating MTurk workers for their work and feedback.

Ethical Considerations

CB2 is a research environment. The focus on a relatively restricted 3D environment reduces the potential for ethical risks. Our use of CB2 has received exemption status by our institution’s IRB office. We recommend that researchers using CB2 obtain IRB approval or exemption for their studies from their institution’s IRB office, or an equivalent body. More broadly, systems that learn from interaction with users raise risks of adopting negative behavior patterns from their users. This is especially an issue in certain contexts, such as open ended conversational interfaces or chatbots. This is an important direction for future work. CB2 can be used to study such adversarial usage scenarios in a relatively safe way.

References

- Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *The IEEE Conference on Computer Vision and Pattern Recognition*.
- Jacob Andreas, Anca Dragan, and Dan Klein. 2017. [Translating neuralese](#). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Yoav Artzi and Luke Zettlemoyer. 2013. [Weakly supervised learning of semantic parsers for mapping instructions to actions](#). *Transactions of the Association of Computational Linguistics*, 1.
- Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A. Knepper, and Yoav Artzi. 2018. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Proceedings of the Robotics: Science and Systems Conference*.
- David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.
- Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*.
- Andrea F Daniele, Mohit Bansal, and Matthew R Walter. 2016. Natural language generation in the context of providing indoor route instructions. In *Proceedings Robotics: Science and Systems Workshop on Model Learning for Human-Robot Communication*.
- Alex Djalali, Sven Lauer, and Christopher Potts. 2012. Corpus evidence for preference-driven interpretation. In *Logic, Language and Meaning*.
- Anna Effenberger, Rhia Singh, Eva Yan, Alane Suhr, and Yoav Artzi. 2021. Analysis of language change in collaborative instruction following. In *Findings of the Association for Computational Linguistics: EMNLP*.
- Daniel Fried, Jacob Andreas, and Dan Klein. 2018. [Unified pragmatic models for generating and following instructions](#). In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42.
- Emiel Hoogeboom, Jorn W.T. Peters, Taco S. Cohen, and Max Welling. 2018. [Hexaconv](#). In *International Conference on Learning Representations*.
- Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. [Learning to execute instructions in a Minecraft dialogue](#). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Julia Kiseleva, Alexey Skrynnik, Artem Zholus, Shrestha Mohanty, Negar Arabzadeh, Marc-Alexandre Côté, Mohammad Aliannejadi, Milagro Teruel, Ziming Li, Mikhail Burtsev, et al. 2022. Iglu 2022: Interactive grounded language understanding in a collaborative environment at neurips 2022. *arXiv preprint arXiv:2205.13771*.
- Noriyuki Kojima, Alane Suhr, and Yoav Artzi. 2021. [Continual learning for grounded instruction generation by observing human following behavior](#). *Transactions of the Association for Computational Linguistics*, 9.
- Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. [Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding](#). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Angeliki Lazaridou and Marco Baroni. 2020. Emergent multi-agent communication in the deep learning era. *ArXiv*, abs/2006.02419.
- Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations*.
- Jason Lee, Kyunghyun Cho, and Douwe Kiela. 2019. Countering language drift via visual grounding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Matthew MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.
- Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. [What to talk about and how? Selective generation using lstms with coarse-to-fine alignment](#). In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. [Collaborative dialogue in Minecraft](#). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

- Christopher Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In *Proceedings of the West Coast Conference on Formal Linguistics*.
- Aaron L Putterman, Kevin Lu, Igor Mordatch, and Pieter Abbeel. 2022. [Pretraining for language conditioned imitation with transformers](#). *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*.
- Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Alane Suhr and Yoav Artzi. 2022. Continual learning for instruction following from realtime feedback. *arXiv preprint arXiv:2212.09710*.
- Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. 2019. [Executing instructions in situated collaborative interactions](#). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*. MIT press.
- Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. 2021. Less is more: Generating grounded navigation instructions from landmarks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Inseq: An Interpretability Toolkit for Sequence Generation Models

Gabriele Sarti  Nils Feldhus  Ludwig Sickert  Oskar van der Wal 

 University of Groningen  University of Amsterdam

 German Research Center for Artificial Intelligence (DFKI), Berlin

g.sarti@rug.nl nils.feldhus@dfki.de l.sickert@outlook.com o.d.vanderwal@uva.nl

Abstract

Past work in natural language processing interpretability focused mainly on popular classification tasks while largely overlooking generation settings, partly due to a lack of dedicated tools. In this work, we introduce Inseq¹, a Python library to democratize access to interpretability analyses of sequence generation models. Inseq enables intuitive and optimized extraction of models' internal information and feature importance scores for popular decoder-only and encoder-decoder Transformers architectures. We showcase its potential by adopting it to highlight gender biases in machine translation models and locate factual knowledge inside GPT-2. Thanks to its extensible interface supporting cutting-edge techniques such as contrastive feature attribution, Inseq can drive future advances in explainable natural language generation, centralizing good practices and enabling fair and reproducible model evaluations.

1 Introduction

Recent years saw an increase in studies and tools aimed at improving our behavioral or mechanistic understanding of neural language models (Belingov and Glass, 2019). In particular, *feature attribution* methods became widely adopted to quantify the importance of input tokens in relation to models' inner processing and final predictions (Madsen et al., 2022b). Many studies applied such techniques to modern deep learning architectures, including Transformers (Vaswani et al., 2017), leveraging gradients (Baehrens et al., 2010; Sundararajan et al., 2017), attention patterns (Xu et al., 2015; Clark et al., 2019) and input perturbations (Zeiler and Fergus, 2014; Feng et al., 2018) to quantify input importance, often leading to controversial outcomes in terms of faithfulness, plausibility and overall usefulness of such explanations (Adebayo

¹Library: <https://github.com/inseq-team/inseq>
Documentation: <https://inseq.readthedocs.io>
This paper describes the Inseq v0.4.0 release on PyPI.

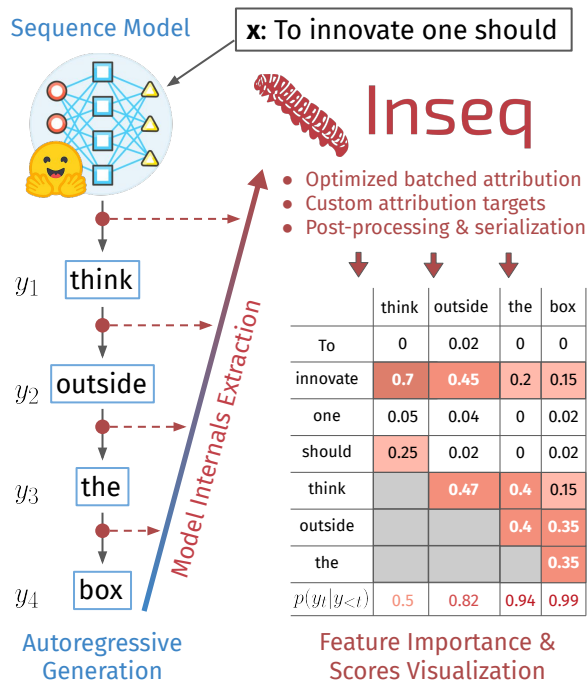



Figure 1: Feature importance and next-step probability extraction and visualization using Inseq with a  Transformers causal language model.

et al., 2018; Jain and Wallace, 2019; Jacovi and Goldberg, 2020; Zafar et al., 2021). However, feature attribution techniques have mainly been applied to classification settings (Atanasova et al., 2020; Wallace et al., 2020; Madsen et al., 2022a; Chrysostomou and Aletras, 2022), with relatively little interest in the more convoluted mechanisms underlying generation. Classification attribution is a single-step process resulting in one importance score per input token, often allowing for intuitive interpretations in relation to the predicted class. Sequential attribution² instead involves a computationally expensive multi-step iteration producing a matrix A_{ij} representing the importance of every input i in the prediction of every generation outcome j (Figure 1). Moreover, since previous

²We use *sequence generation* to refer to all iterative tasks including (but not limited to) natural language generation.

generation steps causally influence following predictions, they must be dynamically incorporated into the set of attributed inputs throughout the process. Lastly, while classification usually involves a limited set of classes and simple output selection (e.g. argmax after softmax), generation routinely works with large vocabularies and non-trivial decoding strategies (Eikema and Aziz, 2020). These differences limited the use of feature attribution methods for generation settings, with relatively few works improving attribution efficiency (Vafa et al., 2021; Ferrando et al., 2022) and explanations’ informativeness (Yin and Neubig, 2022).

In this work, we introduce **Inseq**, a Python library to democratize access to interpretability analyses of generative language models. Inseq centralizes access to a broad set of feature attribution methods, sourced in part from the Captum (Kokhlikyan et al., 2020) framework, enabling a fair comparison of different techniques for all sequence-to-sequence and decoder-only models in the popular 🤗 Transformers library (Wolf et al., 2020). Thanks to its intuitive interface, users can easily integrate interpretability analyses into sequence generation experiments with just 3 lines of code (Figure 2). Nevertheless, Inseq is also highly flexible, including cutting-edge attribution methods with built-in post-processing features (§ 4.1), supporting customizable attribution targets and enabling constrained decoding of arbitrary sequences (§ 4.2). In terms of usability, Inseq greatly simplifies access to local and global explanations with built-in support for a command line interface (CLI), optimized batching enabling dataset-wide attribution, and various methods to visualize, serialize and reload attribution outcomes and generated sequences (§ 4.3). Ultimately, Inseq’s aims to make sequence models first-class citizens in interpretability research and drive future advances in interpretability for generative applications.

2 Related Work

Feature Attribution for Sequence Generation

Work on feature attribution for sequence generation has mainly focused on machine translation (MT). Bahdanau et al. (2015) showed how attention weights of neural MT models encode interpretable alignment patterns. Alvarez-Melis and Jaakkola (2017) adopted a perturbation-based framework to highlight biases in MT systems. Ding et al. (2019); He et al. (2019); Voita et al. (2021a,b) *inter*

```
import inseq

# Load HF Hub model and attribution method
model = inseq.load_model(
    "google/flan-t5-base",
    "integrated_gradients"
)
# Answer and attribute generation steps
attr_out = model.attribute(
    "Does 3 + 3 equal 6?",
    attribute_target=True
)
# Visualize the generated attribution,
# applying default token-level aggregation
attr_out.show()
```

Source Saliency			Prefix Saliency		
	_yes	</s>		_yes	</s>
_Does	0.264	0.153			0.21
_3	0.113	0.099	</s>		
_+	0.096	0.086			
_3	0.096	0.076			
_equal	0.213	0.154			
_6	0.11	0.108			
?	0.106	0.114			

Figure 2: Computing and visualizing source and target-side attributions using Flan-T5 (Chung et al., 2022).

alia conducted analyses on MT word alignments, coreference resolution and training dynamics with various gradient-based attribution methods. Vafa et al. (2021); Ferrando et al. (2022) developed approaches to efficiently compute sequential feature attributions without sacrificing accuracy. Yin and Neubig (2022) introduced contrastive feature attribution to disentangle factors influencing generation in language models. Attribution scores obtained from MT models were also used to detect hallucinatory behavior (Dale et al., 2022; Tang et al., 2022; Xu et al., 2023), providing a compelling practical use case for such explanations.

Tools for NLP Interpretability Although many post-hoc interpretability libraries were released recently, only a few support sequential feature attribution. Notably, LIT (Tenney et al., 2020), a structured framework for analyzing models across modalities, and Ecco (Alammar, 2021), a library specialized in interactive visualizations of model internals. LIT is an all-in-one GUI-based tool to analyze model behaviors on entire datasets. However, the library does not provide out-of-the-box support for 🤗 Transformers models, requiring the definition of custom wrappers to ensure compatibility. Moreover, it has a steep learning curve due to its

advanced UI, which might be inconvenient when working on a small amount of examples. All these factors limit LIT usability for researchers working with custom models, needing access to extracted scores, or being less familiar with interpretability research. On the other hand, Ecco is closer to our work, being based on 🤖 Transformers and having started to support encoder-decoder models concurrently with Inseq development. Despite a marginal overlap in their functionalities, the two libraries provide orthogonal benefits: Inseq’s flexible interface makes it especially suitable for methodical quantitative analyses involving repeated evaluations, while Ecco excels in qualitative analyses aimed at visualizing model internals. Other popular tools such as ERASER (DeYoung et al., 2020), ThermoStat (Feldhus et al., 2021), transformers-interpret (Pierse, 2021) and ferret (Attanasio et al., 2022) do not support sequence models.

3 Design

Inseq combines sequence models sourced from 🤖 Transformers (Wolf et al., 2020) and attribution methods mainly sourced from Captum (Kokhlikyan et al., 2020). While only text-based tasks are currently supported, the library’s modular design³ would enable the inclusion of other modeling frameworks (e.g. fairseq (Ott et al., 2019)) and modalities (e.g. speech) without requiring substantial redesign. Optional dependencies include 🤖 Datasets (Lhoest et al., 2021) and Rich⁴.

3.1 Guiding Principles

Research and Generation-oriented Inseq should support interpretability analyses of a broad set of sequence generation models without focusing narrowly on specific architectures or tasks. Moreover, the inclusion of new, cutting-edge methods should be prioritized to enable fair comparisons with well-established ones.

Scalable The library should provide an optimized interface to a wide range of use cases, models and setups, ranging from interactive attributions of individual examples using toy models to compiling statistics of large language models’ predictions for entire datasets.

Beginner-friendly Inseq should provide built-in access to popular frameworks for sequence genera-

³More details are available in Appendix B.

⁴<https://github.com/Textualize/rich>

	Method	Source	$f(l)$
G	(Input \times) Gradient	Simonyan et al.	✓
	DeepLIFT	Shrikumar et al.	✓
	GradientSHAP	Lundberg and Lee	✗
	Integrated Gradients	Sundararajan et al.	✓
	Discretized IG	Sanyal and Ren	✗
I	Attention Weights	Bahdanau et al.	✓
P	Occlusion (Blank-out)	Zeiler and Fergus	✗
	LIME	Ribeiro et al.	✗
S	(Log) Probability	-	
	Softmax Entropy	-	
	Target Cross-entropy	-	
	Perplexity	-	
	Contrastive Prob. Δ	Yin and Neubig	
	μ MC Dropout Prob.	Gal and Ghahramani	

Table 1: Overview of gradient-based (**G**), internals-based (**I**) and perturbation-based (**P**) attribution methods and built-in step functions (**S**) available in Inseq. $f(l)$ marks methods allowing for attribution of arbitrary intermediate layers.

tion modeling and be fully usable by non-experts at a high level of abstraction, providing sensible defaults for supported attribution methods.

Extensible Inseq should support a high degree of customization for experienced users, with out-of-the-box support for user-defined solutions to enable future investigations into models’ behaviors.

4 Modules and Functionalities

4.1 Feature Attribution and Post-processing

At its core, Inseq provides a simple interface to apply feature attribution techniques for sequence generation tasks. We categorize methods in three groups, *gradient-based*, *internals-based* and *perturbation-based*, depending on their underlying approach to importance quantification.⁵ Table 1 presents the full list of supported methods. Aside from popular model-agnostic methods, Inseq notably provides built-in support for attention weight attribution and the cutting-edge Discretized Integrated Gradients method (Sanyal and Ren, 2021). Moreover, multiple methods allow for the importance attribution of custom intermediate model layers, simplifying studies on representational structures and information mixing in sequential models, such as our case study of Section 5.2.

Source and target-side attribution When using encoder-decoder architectures, users can set the

⁵We distinguish between gradient- and internals-based methods to account for their difference in scores’ granularity.

`attribute_target` parameter to include or exclude the generated prefix in the attributed inputs. In most cases, this should be desirable to account for recently generated tokens when explaining model behaviors, such as when to terminate the generation (e.g. relying on the presence `_yes` in the target prefix to predict `</s>` in Figure 2, bottom-right matrix). However, attributing the source side separately could prove useful, for example, to derive word alignments from importance scores.

Post-processing of attribution outputs Aggregation is a fundamental but often overlooked step in attribution-based analyses since most methods produce neuron-level or subword-level importance scores that would otherwise be difficult to interpret. Inseq includes several `Aggregator` classes to perform attribution aggregation across various dimensions. For example, the input word “Explanation” could be tokenized in two subword tokens “Expl” and “anation”, and each token would receive N importance scores, with N being the model embedding dimension. In this case, aggregators could first merge subword-level scores into word-level scores, and then merge granular embedding-level scores to obtain a single token-level score that is easier to interpret. Moreover, aggregation could prove especially helpful for long-form generation tasks such as summarization, where word-level importance scores could be aggregated to obtain a measure of sentence-level relevance. Notably, Inseq allows chaining multiple aggregators like in the example above using the `AggregatorPipeline` class, and provides a `PairAggregator` to aggregate different attribution maps, simplifying the conduction of contrastive analyses as in Section 5.1.⁶

4.2 Customizing generation and attribution

During attribution, Inseq first generates target tokens using 🤖 Transformers and then attributes them step by step. If a custom target string is specified alongside model inputs, the generation step is instead skipped, and the provided text is attributed by constraining the decoding of its tokens⁷. Constrained attribution can be used, among other things, for contrastive comparisons of minimal pairs and to obtain model justifications for desired outputs.

⁶See Appendix C for an example.

⁷Constrained decoding users should be aware of its limitations in the presence of a high distributional discrepancy with natural model outputs (Vamvas and Sennrich, 2021).

Custom step functions At every attribution step, Inseq can use models’ internal information to extract scores of interest (e.g. probabilities, entropy) that can be useful, among other things, to quantify model uncertainty (e.g. how likely the generated `_yes` token was given the context in Figure 2). Inseq provides access to multiple built-in step functions (Table 1, S) enabling the computation of these scores, and allows users to create and register new custom ones. Step scores are computed together with the attribution, returned as separate sequences in the output, and visualized alongside importance scores (e.g. the $p(y_t|y_{<t})$ row in Figure 1).

Step functions as attribution targets For methods relying on model outputs to predict input importance (gradient and perturbation-based), feature attributions are commonly obtained from the model’s output logits or class probabilities (Bastings et al., 2022). However, recent work showed the effectiveness of using targets such as the probability difference of a contrastive output pair to answer interesting questions like “What inputs drive the prediction of y rather than \hat{y} ?” (Yin and Neubig, 2022). In light of these advances, Inseq users can leverage any built-in or custom-defined step function as an attribution target, enabling advanced use cases like contrastive comparisons and uncertainty-weighted attribution using MC Dropout (Gal and Ghahramani, 2016).

4.3 Usability Features

Batched and span-focused attributions The library provides built-in batching capabilities, enabling users to go beyond single sentences and attribute even entire datasets in a single function call. When the attribution of a specific span of interest is needed, Inseq also allows specifying a start and end position for the attribution process. This functionality greatly accelerates the attribution process for studies on localized phenomena (e.g. pronoun coreference in MT models).

CLI, Serialization and Visualization The Inseq library offers an API to attribute single examples or entire 🤖 Datasets from the command line and save resulting outputs and visualizations to a file. Attribution outputs can be saved and loaded in JSON format with their respective metadata to easily identify the provenance of contents. Attributions can be visualized in the command line or IPython notebooks and exported as HTML files.

Quantized Model Attribution Supporting the attribution of large models is critical given recent scaling tendencies (Kaplan et al., 2020). All models allowing for quantization using bitsandbytes (Dettmers et al., 2022) can be loaded in 8-bit directly from 🍷 Transformers, and their attributions can be computed normally using Inseq.⁸ A minimal manual evaluation of 8-bit attribution outputs for Section 5.2 study shows minimal discrepancies compared to full-precision results.

5 Case Studies

5.1 Gender Bias in Machine Translation

In the first case study, we use Inseq to investigate gender bias in MT models. Studying social biases embedded in these models is crucial to understand and mitigate the representational and allocative harms they might engender (Blodgett et al., 2020). Savoldi et al. (2021) note that the study of bias in MT could benefit from explainability techniques to identify spurious cues exploited by the model and the interaction of different features that can lead to intersectional bias.

Synthetic Setup: Turkish to English The Turkish language uses the gender-neutral pronoun *o*, which can be translated into English as either “he”, “she”, or “it”, making it interesting to study gender bias in MT when associated with a language such as English for which models will tend to choose a gendered pronoun form. Previous works leveraged translations from gender-neutral languages to show gender bias present in translation systems (Cho et al., 2019; Prates et al., 2020; Farkas and Németh, 2022). We repeat this simple setup using a Turkish-to-English MarianMT model (Tiedemann, 2020) and compute different metrics to quantify gender bias using Inseq.

We select 49 Turkish occupation terms verified by a native speaker (see Appendix E) and use them to infill the template sentence “*O bir ___*” (He/She is a(n) ___). For each translation, we compute attribution scores for source Turkish pronoun (x_{pron}) and occupation (x_{occ}) tokens⁹ when generating the target English pronoun (y_{pron}) using Integrated Gradients (IG), Gradients (∇), and Input \times Gradient ($I \times G$),¹⁰. We also collect target pronoun probabili-

⁸bitsandbytes 0.37.0 required for backward method, see Appendix D for an example.

⁹For multi-token occupation terms, e.g., *bilim insanı* (scientist), the attribution score of the first token was used.

¹⁰We set approx. steps to ensure convergence $\Delta < 0.05$

	Base		$\text{♀} \rightarrow \text{♂}$	
	x_{pron}	x_{occ}	x_{pron}	x_{occ}
$p(y_{\text{pron}})$	0.01		-0.44*	
∇	-0.16	0.25*	0.23*	-0.00
IG	-0.08	0.09	0.11	0.17
$I \times G$	-0.11	0.22*	0.22*	-0.01

Table 2: **Gender Bias in Turkish-to-English MT:** Kendall’s τ correlation of MT model metrics with U.S. labor statistics. * = Significant correlation ($p < .05$).

ties ($p(y_{\text{pron}})$), rank the 49 occupation terms using these metrics, and finally compute Kendall’s τ correlation with the percentage of women working in the respective fields, using U.S. labor statistics as in previous works (e.g., Caliskan et al., 2017; Rudinger et al., 2018). Table 2 presents our results.

In the **base case**, we correlate the different metrics with how much the gender distribution deviates from an equal distribution (50 – 50%) for each occupation (i.e., the gender bias irrespective of the direction). We observe a strong gender bias, with “she” being chosen only for 5 out of 49 translations and gender-neutral variants never being produced by the MT model. We find a low correlation between pronoun probability and the degree of gender stereotype associated with the occupation. Moreover, we note a weaker correlation for IG compared to the other two methods. For those, attribution scores for x_{occ} show significant correlations with labor statistics, supporting the intuition that the MT model will accord higher importance to source occupation terms associated to gender-stereotypical occupations when predicting the gendered target pronoun.

In the **gender-swap case** ($\text{♀} \rightarrow \text{♂}$), we use the PairAggregator class to contrastively compare attribution scores and probabilities when translating the pronoun as “She” or “He”.¹¹ We correlate resulting scores with the % of women working in the respective occupation and find strong correlations for $p(y_{\text{pron}})$, supporting the validity of contrastive approaches in uncovering gender bias.

Qualitative Example: English to Dutch We qualitatively analyze biased MT outputs, showing how attributions can help develop hypotheses about models’ behavior. Table 3 (top) shows the $I \times G$ attributions for English-to-Dutch translation using M2M-100 (418M, Fan et al., 2021). The model

for IG. All methods use the L2 norm to obtain token-level attributions.

¹¹An example is provided in Appendix C.

Source	De	leraar	verliest	zijn	baan
The teacher	0.10	0.08	0.04	0.03	0.02
loses her job	0.11	0.20	0.06	0.03	0.05
	0.11	0.09	0.25	0.07	0.07
	0.15	0.09	0.10	0.21	0.07
	0.10	0.08	0.08	0.10	0.24
Target	De	leraar	verliest	zijn	baan
De leraar		0.23	0.05	0.06	0.04
verliest zijn			0.17	0.13	0.03
				0.18	0.08
					0.26
$p(y_t)$	0.69	0.28	0.35	0.65	0.29
Source	De	$\sigma \rightarrow \circ$	verliest	haar	baan
The teacher	0.00	-0.02	0.00	0.00	0.00
loses her job	0.00	-0.05	-0.01	-0.01	-0.01
	0.00	-0.02	-0.01	-0.02	-0.01
	0.00	-0.01	-0.01	-0.10	0.01
	0.00	-0.02	-0.01	-0.02	-0.02
Target	De	$\sigma \rightarrow \circ$	verliest	haar	baan
De $\sigma \rightarrow \circ$		-0.07	-0.01	0.01	-0.01
verliest haar			0.09	0.18	0.02
				-0.03	0.00
					0.00
$\Delta p(y_t)$	0.00	-0.23	0.13	0.20	0.00

Table 3: **Top:** Attribution of pronoun gender mistranslation using M2M-100. **Bottom:** Target attribution difference when swapping the target noun gender ($\sigma \rightarrow \circ$) from *leraar* (male) to *leerkracht* (gender-neutral).

mistranslates the pronoun “her” into the masculine form *zijn* (his). We find that the wrongly translated pronoun exhibits high probability but does not associate substantial importance to the source occupation term “teacher”. Instead, we find good relative importance for the preceding word and *leraar* (male teacher). This suggests a strong prior bias for masculine variants, shown by the pronoun *zijn* and the noun *leraar*, as a possible cause for this mistranslation. When considering the contrastive example obtained by swapping *leraar* with its gender-neutral variant *leerkracht* (Table 3, bottom), we find increased importance of the target occupation in determining the correctly-gendered target pronoun *haar* (her). Our results highlight the tendency of MT models to attend inputs sequentially rather than relying on context, hinting at the known benefits of context-aware models for pronoun translation (Voita et al., 2018).

5.2 Locating Factual Knowledge inside GPT-2 with Contrastive Attribution Tracing

For our second case study, we experiment with a novel attribution-based technique to locate factual knowledge encoded in the layers of GPT-2 1.5B (Radford et al., 2019). Specifically, we aim to reproduce the results of Meng et al. (2022), showing the influence of intermediate layers in mediat-

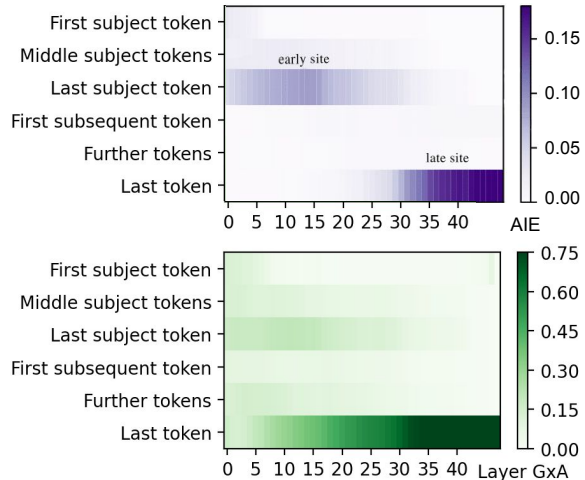


Figure 3: **Top:** Estimated causal importance of GPT-2 XL layers for predicting factual associations, as reported by Meng et al. (2022). **Bottom:** Average GPT-2 XL Gradient \times Layer Activation scores obtained with Inseq using contrastive factual pairs as attribution targets.

ing the recall of factual statements such as ‘*The Eiffel Tower is located in the city of* \rightarrow *Paris*’. Meng et al. (2022) estimate the effect of network components in the prediction of factual statements as the difference in probability of a correct target (e.g. *Paris*), given a corrupted subject embedding (e.g. for *Eiffel Tower*), before and after restoring clean activations for some input tokens at different layers of the network. Apart from the obvious importance of final token states in terminal layers, their results highlight the presence of an early site associated with the last subject token playing an important role in recalling the network’s factual knowledge (Figure 3, top).

To verify such results, we propose a novel knowledge location method, which we name Contrastive Attribution Tracing (CAT), adopting the contrastive attribution paradigm of Yin and Neubig (2022) to locate relevant network components by attributing minimal pairs of correct and wrong factual targets (e.g. *Paris* vs. *Rome* for the example above). To perform the contrastive attribution, we use the Layer Gradient \times Activation method, a layer-specific variant of Input \times Gradient, to propagate gradients up to intermediate network activations instead of reaching input tokens. The resulting attribution scores hence answer the question “How important are layer L activations for prefix token t in predicting the correct factual target over a wrong one?”. We compute attribution scores for 1000 statements taken from the Counterfact Statement dataset (Meng et al., 2022) and present

averaged results in Figure 3 (bottom).¹² Our results closely match those of the original authors, providing further evidence of how attribution methods can be used to identify salient network components and guide model editing, as shown by Dai et al. (2022) and Nanda (2023).

To our best knowledge, the proposed CAT method is the most efficient knowledge location technique to date, requiring only a single forward and backward pass of the attributed model. Patching-based approaches such as causal mediation (Meng et al., 2022), on the other hand, provide causal guarantees of feature importance at the price of being more computationally intensive. Despite lacking the causal guarantees of such methods, CAT can provide an approximation of feature importance and greatly simplify the study of knowledge encoded in large language model representations thanks to its efficiency.

6 Conclusion

We introduced Inseq, an easy-to-use but versatile toolkit for interpreting sequence generation models. With many libraries focused on the study of classification models, Inseq is the first tool explicitly aimed at analyzing systems for tasks such as machine translation, code synthesis, and dialogue generation. Researchers can easily add interpretability evaluations to their studies using our library to identify unwanted biases and interesting phenomena in their models’ predictions. We plan to provide continued support and explore developments for Inseq,¹³ to provide simple and centralized access to a comprehensive set of thoroughly-tested implementations for the interpretability community. In conclusion, we believe that Inseq has the potential to drive real progress in explainable language generation by accelerating the development of new analysis techniques, and we encourage members of this research field to join our development efforts.

Acknowledgments

We thank Ece Takmaz for verifying the Turkish word list used in Section 5.1. GS and AB acknowledge the support of the Dutch Research Council (NWO) as part of the project InDeep (NWA.1292.19.399). NF is supported by the German Federal Ministry of Education and Research as part of the project XAINES (01IW20005). OvdW’s

contributions are financed by the NWO as part of the project “The biased reality of online media – Using stereotypes to make media manipulation visible” (406.DI.19.059).

Broader Impact and Ethics Statement

Reliability of Attribution Methods The plausibility and faithfulness of attribution methods supported by Inseq is an active matter of debate in the research community, without clear-cut guarantees in identifying specific model behaviors, and prone to users’ own biases (Jacovi and Goldberg, 2020). We emphasize that explanations produced with Inseq should not be adopted in high-risk and user-facing contexts. We encourage Inseq users to critically approach results obtained from our toolkit and validate them on a case-by-case basis.

Technical Limitations and Contributions

While Inseq greatly simplifies comparisons across different attribution methods to ensure their mutual consistency, it does not provide explicit ways of evaluating the quality of produced attributions in terms of faithfulness or plausibility. Moreover, many recent methods still need to be included due to the rapid pace of interpretability research in natural language processing and the small size of our development team. To foster an open and inclusive development environment, we encourage all interested users and new methods’ authors to contribute to the development of Inseq by adding their interpretability methods of interest.

Gender Bias Case Study The case study of Section 5.1 assumes a simplified concept of binary gender to allow for a more straightforward evaluation of the results. However, we encourage other researchers to consider non-binary gender and different marginalized groups in future bias studies. We acknowledge that measuring bias in language models is complex and that care must be taken in its conceptualization and validation (Blodgett et al., 2020; van der Wal et al., 2022; Bommasani and Liang, 2022), even more so in multilingual settings (Talat et al., 2022). For this reason, we do not claim to provide a definite bias analysis of these MT models – especially in light of the aforementioned attributions’ faithfulness issues. The study’s primary purpose is to demonstrate how attribution methods could be used for exploring social biases in sequence-to-sequence models and showcase the related Inseq functionalities.

¹²Figure 6 of Appendix D presents some examples.

¹³Planned developments available in Appendix F.

References

- Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Y. Zou. 2019. [Gradio: Hassle-free sharing and testing of ML models in the wild](#). *ArXiv*, abs/1906.02569.
- Samira Abnar and Willem Zuidema. 2020. [Quantifying attention flow in transformers](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.
- Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. [Sanity checks for saliency maps](#). In *Advances in Neural Information Processing Systems*, volume 31, pages 9505–9515, Montréal, Canada. Curran Associates, Inc.
- J Alammar. 2021. [Ecco: An open source library for the explainability of transformer language models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online. Association for Computational Linguistics.
- David Alvarez-Melis and Tommi Jaakkola. 2017. [A causal framework for explaining the predictions of black-box sequence-to-sequence models](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421, Copenhagen, Denmark. Association for Computational Linguistics.
- Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. [A diagnostic study of explainability techniques for text classification](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.
- Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. 2022. [ferret: a framework for benchmarking explainers on transformers](#). *ArXiv*, abs/2208.01575.
- Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. [On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation](#). *PLOS ONE*, 10(7):1–46.
- David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. [How to explain individual classification decisions](#). *J. Mach. Learn. Res.*, 11:1803–1831.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.
- Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. [“will you find these shortcuts?” a protocol for evaluating the faithfulness of input saliency methods for text classification](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yonatan Belinkov and James Glass. 2019. [Analysis methods in neural language processing: A survey](#). *Transactions of the Association for Computational Linguistics*, 7:49–72.
- Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. [Language \(technology\) is power: A critical survey of “bias” in NLP](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.
- Rishi Bommasani and Percy Liang. 2022. [Trustworthy social bias measurement](#). *ArXiv*, abs/2212.11672.
- Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. [Semantics derived automatically from language corpora contain human-like biases](#). *Science*, 356(6334):183–186.
- Won Ik Cho, Ji Won Kim, Seok Min Kim, and Nam Soo Kim. 2019. [On measuring gender bias in translation of gender-neutral pronouns](#). In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 173–181, Florence, Italy. Association for Computational Linguistics.
- George Chrysostomou and Nikolaos Aletras. 2022. [An empirical study on explanations in out-of-domain settings](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6920–6938, Dublin, Ireland. Association for Computational Linguistics.
- Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc Le, and Jason Wei. 2022. [Scaling instruction-finetuned language models](#). *ArXiv*, abs/2210.11416.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. [What does BERT look at? an analysis of BERT’s attention](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. [Knowledge neurons in pretrained transformers](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.
- David Dale, Elena Voita, Loïc Barrault, and Marta Ruiz Costa-jussà. 2022. [Detecting and mitigating hallucinations in machine translation: Model internal workings alone do well, sentence similarity even better](#). *ArXiv*, abs/2212.08597.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. [GPT3.int8\(\): 8-bit matrix multiplication for transformers at scale](#). In *Advances in Neural Information Processing Systems*.
- Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. [ERASER: A benchmark to evaluate rationalized NLP models](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.
- Shuoyang Ding, Hainan Xu, and Philipp Koehn. 2019. [Saliency-driven word alignment interpretation for neural machine translation](#). In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 1–12, Florence, Italy. Association for Computational Linguistics.
- Bryan Eikema and Wilker Aziz. 2020. [Is MAP decoding all you need? the inadequacy of the mode in neural machine translation](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Man-deep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2021. [Beyond english-centric multilingual machine translation](#). *J. Mach. Learn. Res.*, 22(1).
- Anna Farkas and Renáta Németh. 2022. [How to measure gender bias in machine translation: Real-world oriented machine translators, multiple reference points](#). *Social Sciences & Humanities Open*, 5(1):100239.
- Nils Feldhus, Robert Schwarzenberg, and Sebastian Möller. 2021. [Thermostat: A large collection of NLP model explanations and analysis tools](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 87–95, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. [Pathologies of neural models make interpretations difficult](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.
- Javier Ferrando, Gerard I. Gállego, Belen Alastruey, Carlos Escolano, and Marta R. Costa-jussà. 2022. [Towards opening the black box of neural machine translation: Source and target interpretations of the transformer](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8756–8769, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Javier Ferrando, Gerard I. Gállego, Ioannis Tsiamas, and Marta Ruiz Costa-jussà. 2023. [Explaining how transformers use context to build predictions](#). *ArXiv*, abs/2305.12535.
- Yarin Gal and Zoubin Ghahramani. 2016. [Dropout as a bayesian approximation: Representing model uncertainty in deep learning](#). In *Proceedings of The 33rd International Conference on Machine Learning Research*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, NY, USA. Proceedings of Machine Learning Research (PLMR).
- Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu, and Shuming Shi. 2019. [Towards understanding neural machine translation with word importance](#). 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 953–962, Hong Kong, China. Association for Computational Linguistics.
- Alon Jacovi and Yoav Goldberg. 2020. [Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.
- Sarthak Jain, Varun Manjunatha, Byron Wallace, and Ani Nenkova. 2022. [Influence functions for sequence tagging models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 824–839, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Sarthak Jain and Byron C. Wallace. 2019. [Attention is not Explanation](#). 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.
- Zhiying Jiang, Raphael Tang, Ji Xin, and Jimmy Lin. 2020. [Inserting Information Bottlenecks for Attribution in Transformers](#). In *Findings of the Association*

- for *Computational Linguistics: EMNLP 2020*, pages 3850–3857, Online. Association for Computational Linguistics.
- Andrei Kapishnikov, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. 2021. [Guided integrated gradients: An adaptive path method for removing noise](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5050–5058.
- Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *ArXiv*, abs/2001.08361.
- Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. [Attention is not only a weight: Analyzing transformers with vector norms](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.
- Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. [Incorporating Residual and Normalization Layers into Analysis of Masked Language Models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. [Feed-forward blocks control contextualization in masked language models](#). *ArXiv*, abs/2302.00456.
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. [Captum: A unified and generic model interpretability library for pytorch](#). *ArXiv*, abs/2009.07896.
- Tsz Kin Lam, Eva Hasler, and Felix Hieber. 2022. [Analyzing the use of influence functions for instance-specific data filtering in neural machine translation](#). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 295–309, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Shahar Levy, Koren Lazar, and Gabriel Stanovsky. 2021. [Collecting a large-scale gender bias dataset for coreference resolution and machine translation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2470–2480, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. [Datasets: A community library for natural language processing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Scott M. Lundberg and Su-In Lee. 2017. [A unified approach to interpreting model predictions](#). In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30, page 4768–4777, Long Beach, California, USA. Curran Associates Inc.
- Andreas Madsen, Nicholas Meade, Vaibhav Adlakha, and Siva Reddy. 2022a. [Evaluating the faithfulness of importance measures in NLP by recursively masking allegedly important tokens and retraining](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1731–1751, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022b. [Post-hoc interpretability for neural nlp: A survey](#). *ACM Comput. Surv.*, 55(8).
- Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in GPT](#). In *Advances in Neural Information Processing Systems*.
- Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. [GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.
- Hosein Mohebbi, Willem H. Zuidema, Grzegorz Chrupała, and A. Alishahi. 2023. [Quantifying context mixing in transformers](#). *ArXiv*, abs/2301.12971.
- Neel Nanda. 2023. [Attribution patching: Activation patching at industrial scale](#). Blog post.
- 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 (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

- Charles Pierson. 2021. [Transformers interpret](#). Python library.
- Marcelo OR Prates, Pedro H Avelar, and Luís C Lamb. 2020. Assessing gender bias in machine translation: a case study with google translate. *Neural Computing and Applications*, 32:6363–6381.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#). *OpenAI Blog*.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. ["why should i trust you?": Explaining the predictions of any classifier](#). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, page 1135–1144, New York, NY, USA. Association for Computing Machinery.
- Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. [Gender bias in coreference resolution](#). 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 8–14, New Orleans, Louisiana. Association for Computational Linguistics.
- Soumya Sanyal and Xiang Ren. 2021. [Discretized integrated gradients for explaining language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10285–10299, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Beatrice Savoldi, Marco Gaido, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. [Gender bias in machine translation](#). *Transactions of the Association for Computational Linguistics*, 9:845–874.
- Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. [Learning important features through propagating activation differences](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR.
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. [Deep inside convolutional networks: Visualising image classification models and saliency maps](#). *arXiv*.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. [Axiomatic attribution for deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, page 3319–3328. Journal of Machine Learning Research (JMLR).
- Zeeraq Talat, Aurélie Névél, Stella Biderman, Miruna Clinciu, Manan Dey, Shayne Longpre, Sasha Lucioni, Maraim Masoud, Margaret Mitchell, Dragomir Radev, Shanya Sharma, Arjun Subramonian, Jaesung Tae, Samson Tan, Deepak Tunuguntla, and Oskar Van Der Wal. 2022. [You reap what you sow: On the challenges of bias evaluation under multilingual settings](#). In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 26–41, virtual+Dublin. Association for Computational Linguistics.
- Joel Tang, M. Fomicheva, and Lucia Specia. 2022. [Reducing hallucinations in neural machine translation with feature attribution](#). *ArXiv*, abs/2211.09878.
- Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. [The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.
- Jörg Tiedemann. 2020. [The tatoeba translation challenge – realistic data sets for low resource and multilingual MT](#). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.
- Keyon Vafa, Yuntian Deng, David Blei, and Alexander Rush. 2021. [Rationales for sequential predictions](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10314–10332, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jannis Vamvas and Rico Sennrich. 2021. [On the limits of minimal pairs in contrastive evaluation](#). In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 58–68, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Oskar van der Wal, Dominik Bachmann, Alina Leidinger, Leendert van Maanen, Willem Zuidema, and Katrin Schulz. 2022. [Undesirable biases in nlp: Averting a crisis of measurement](#). *ArXiv*, abs/2211.13709.
- 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.
- Elena Voita, Rico Sennrich, and Ivan Titov. 2021a. [Analyzing the source and target contributions to predictions in neural machine translation](#). 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 1126–1140, Online. Association for Computational Linguistics.
- Elena Voita, Rico Sennrich, and Ivan Titov. 2021b. [Language modeling, lexical translation, reordering: The](#)

- training process of NMT through the lens of classical SMT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8478–8491, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. [Context-aware neural machine translation learns anaphora resolution](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.
- Eric Wallace, Matt Gardner, and Sameer Singh. 2020. [Interpreting predictions of NLP models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 20–23, Online. 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.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. [Show, attend and tell: Neural image caption generation with visual attention](#). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.
- Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna J. Martindale, and Marine Carpuat. 2023. [Understanding and detecting hallucinations in neural machine translation via model introspection](#). *ArXiv*, abs/2301.07779.
- Kayo Yin and Graham Neubig. 2022. [Interpreting language models with contrastive explanations](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Muhammad Bilal Zafar, Michele Donini, Dylan Slack, Cedric Archambeau, Sanjiv Das, and Krishnamurthy Kenthapadi. 2021. [On the lack of robust interpretability of neural text classifiers](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3730–3740, Online. Association for Computational Linguistics.
- Matthew D. Zeiler and Rob Fergus. 2014. [Visualizing and understanding convolutional networks](#). In *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

A Authors’ Contributions

Authors jointly contributed to the writing and revision of the paper.

Gabriele Sarti Organized and led the project, developed the first public release of the Inseq library, conducted the case study of Section 5.2.

Nils Feldhus Implemented the perturbation-based methods in Inseq and contributed to the validation of the case study of Section 5.2.

Ludwig Sickert Implemented the attention-based attribution method in Inseq.

Oskar van der Wal Conducted the experiments in the gender bias case study of Section 5.1.

Malvina Nissim and **Arianna Bisazza** ensured the soundness of the overall process and provided valuable inputs for the initial design of the toolkit.

B Additional Design Details

Figure 4 presents the Inseq hierarchy of models and attribution methods. The model-method connection enables out-of-the-box attribution using the selected method. Framework-specific and architecture-specific classes enable extending Inseq to new modeling architectures and frameworks.

C Example of Pair Aggregation for Contrastive MT Comparison

An example of gender translation pair using the synthetic template of Section 5.1 is show in Figure 5, highlighting a large drop in probability when switching the gendered pronoun for highly gender-stereotypical professions, similar to Table 2 results.

D Example of Quantized Contrastive Attribution of Factual Knowledge

Figure 6 presents code used in Section 5.2 case study, with visualized attribution scores for contrastive examples in the evaluated dataset.

E Gender Bias in Machine Translation

Table 4 shows the list of occupation terms used in the gender bias case study (Section 5.1). We correlate the ranking of occupations based on the selected attribution metrics and probabilities with U.S. labor statistics¹⁴ (bls_pct_female column). Table 3 example was taken from the BUG dataset (Levy et al., 2021).

¹⁴<https://github.com/rudinger/winogender-schemas>

Turkish	English	Turkish	English
teknisyen	technician	memur	officer
muhasebeci	accountant	patolog	pathologist
süpervizör	supervisor	öğretmen	teacher
mühendis	engineer	avukat	lawyer
işçi	worker	planlamacı	planner
eğitimci	educator	yönetici	practitioner
katip	clerk	tesisatçı	plumber
danışman	consultant	eğitmen	instructor
müfettiş	inspector	cerrah	surgeon
tamirci	mechanic	veteriner	veterinarian
müdür	manager	kimyager	chemist
terapist	therapist	makinist	machinist
resepsiyonist	receptionist	mimar	architect
kütüphaneci	librarian	kuaför	hairstresser
ressam	painter	fırıncı	baker
eczacı	pharmacist	programlamacı	programmer
kapıcı	janitor	itfaiyeci	firefighter
psikolog	psychologist	bilim insanı	scientist
doktor	physician	sevk memuru	dispatcher
marangoz	carpenter	kasiyer	cashier
hemşire	nurse	komisyoncu	broker
araştırmacı	investigator	şef	chef
barmen	bartender	doktor	doctor
uzman	specialist	sekreter	secretary
elektrikçi	electrician		

Table 4: List of the 49 Turkish occupation terms and their English translations used in the gender bias case study (Section 5.1).

	Method	Source
G	Guided Integrated Gradients	Kapishnikov et al.
	LRP	Bach et al.
I	Attention Rollout & Flow	Abnar and Zuidema
	Attention × Vector Norm	Kobayashi et al.
	Attention × Attn. Block Norm	Kobayashi et al.
	GlobEnc	Modarressi et al.
	ALTI+	Ferrando et al.
	Attention × Trans. Block Norm	Kobayashi et al.
	ALTI-Logit	Ferrando et al.
P	Information Bottlenecks	Jiang et al.
	Value Zeroing	Mohebbi et al.
	Input Reduction	Feng et al.
	Activation Patching	Meng et al.

Table 5: Gradient-based (**G**), internals-based (**I**) and perturbation-based (**P**) attribution methods for which we plan to include support in future Inseq releases.

F Planned Developments and Next Steps

We plan to continuously expand the core functionality of the library by adding support for a wider range of attribution methods. Table 5 shows a subset of methods we consider including in future releases. Besides new methods, we also intend to significantly improve result visualization using an interactive interface backed by Gradio Blocks (Abid et al., 2019), work on interoperability features together with ferret developers (Attanasio et al., 2022) to simplify the evaluation of sequence attributions, and include sequential instance attribution methods (Lam et al., 2022; Jain et al., 2022) for training data attribution.

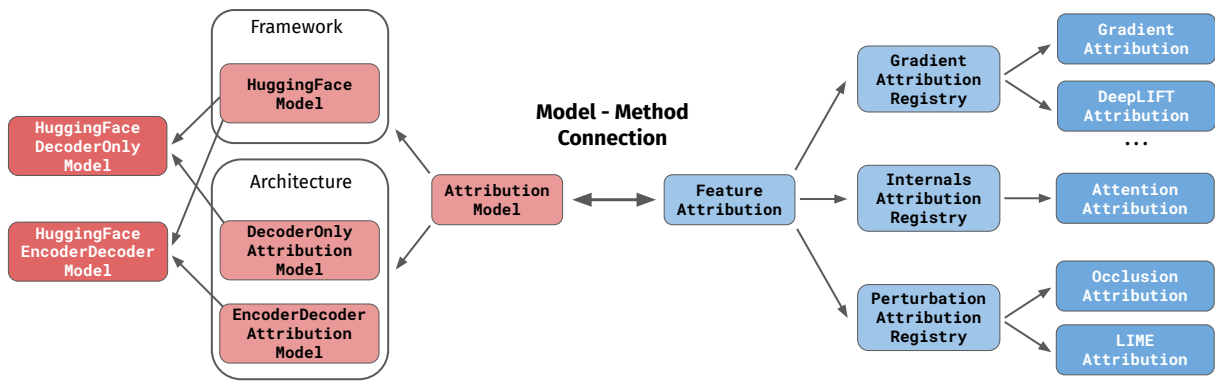


Figure 4: Inseq models and attribution methods. **Concrete classes** combine abstract **framework** and **architecture** attribution models classes, and are derived from abstract attribution methods' **categories**.

```

import inseq
from inseq.data.aggregator import AggregatorPipeline, SubwordAggregator,
    SequenceAttributionAggregator, PairAggregator

# Load the TR-EN translation model and attach the IG method
model = inseq.load_model("Helsinki-NLP/opus-mt-tr-en", "integrated_gradients")

# Batch attribute with forced decoding. Return probabilities, no target attr.
out = model.attribute(
    ["0 bir teknisyen", "0 bir teknisyen"],
    ["She is a technician.", "He is a technician."],
    step_scores=["probability"],
    # The following attributes are specific to the IG method
    internal_batch_size=100,
    n_steps=300
)

# Aggregation pipeline composed by two steps:
# 1. Aggregate subword tokens across all dimensions: [l1, l2, dim] -> [l3, l4, dim]
# 2. Aggregate hidden size to produce token-level attributions: [l1, l2, dim] -> [l1, l2]
subw_aggregator = AggregatorPipeline([SubwordAggregator, SequenceAttributionAggregator])

# Aggregate attributions using the pipeline
masculine = out.sequence_attributions[0].aggregate(aggregator=subw_aggregator)
feminine = out.sequence_attributions[1].aggregate(aggregator=subw_aggregator)

# Take the diff of the scores of the two attributions, show it and return the HTML
html = masculine.show(aggregator=PairAggregator, paired_attr=feminine, return_html=True)

```

Source Saliency Heatmap
x: Generated tokens, y: Attributed tokens

	_She → _He	_is	_a	_technician.	</s>
_O	0.115	-0.004	0.011	0.003	0.014
_bir	0.069	-0.023	-0.019	-0.006	-0.015
_teknisyen	-0.184	0.027	0.008	0.003	0.001
</s>	0.0	0.0	0.0	0.0	0.0
probability	0.46	0.004	0.003	-0.014	0.001

Figure 5: Comparing attributions for a synthetic Turkish-to-English translation example with underspecified source pronoun gender using a MarianMT Turkish-to-English translation model (Tiedemann, 2020). Values in the visualized attribution matrix show a 46% higher probability of producing the masculine pronoun in the translation and a relative decrease of 18.4% in the importance of the Turkish occupation term compared to the feminine pronoun case.

```

import inseq
from datasets import load_dataset
from transformers import AutoModelForCausalLM, AutoTokenizer

# The model is loaded in 8-bit on available GPUs
model = AutoModelForCausalLM.from_pretrained("gpt2-xl", load_in_8bit=True, device_map="auto")
tokenizer = AutoTokenizer.from_pretrained("gpt2-xl")
# Counterfact datasets used by Meng et al. (2022)
data = load_dataset("NeelNanda/counterfact-tracing")["train"]

# GPT-2 XL is a Transformer model with 48 layers
for layer in range(48):
    attrib_model = inseq.load_model(
        model,
        "layer_gradient_x_activation",
        tokenizer="gpt2-xl",
        target_layer=model.transformer.h[layer].mlp,
    )
    for i, ex in data:
        # e.g. "The capital of Second Spanish Republic is"
        prompt = ex["relation"].format(ex["subject"])
        # e.g. "The capital of Second Spanish Republic is Madrid"
        true_answer = prompt + ex["target_true"]
        # e.g. "The capital of Second Spanish Republic is Paris"
        false_answer = prompt + ex["target_false"]
        contrast = attrib_model.encode(false_answer)
        # Contrastive attribution of true vs false answer
        out = attrib_model.attribute(
            prompt,
            true_answer,
            attributed_fn="contrast_prob_diff",
            contrast_ids=contrast.input_ids,
            contrast_attention_mask=contrast.attention_mask,
            step_scores=["contrast_prob_diff"],
            show_progress=False,
        )
# Aggregation and plotting omitted for brevity
...

```

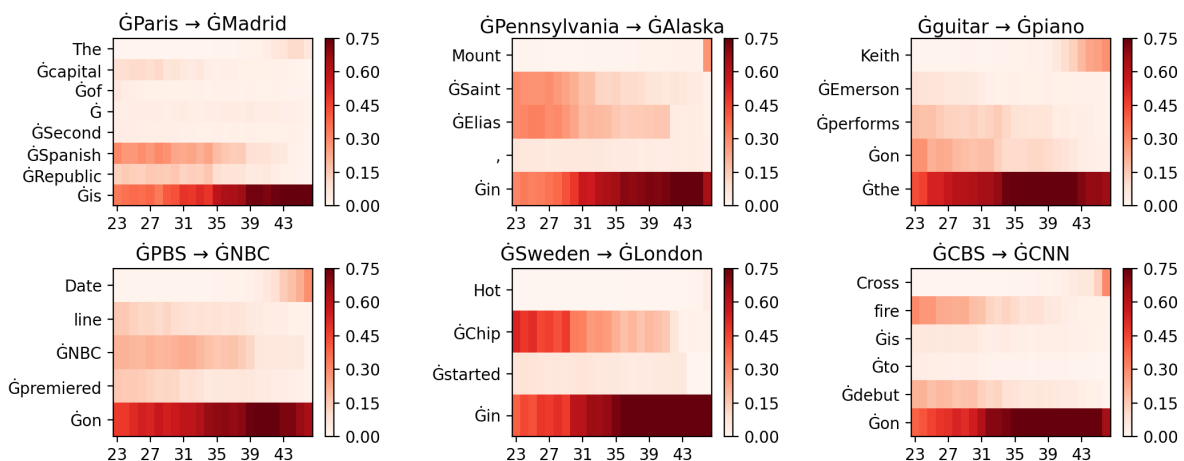


Figure 6: **Top:** Example code to contrastively attribute factual statements from the Counterfact Tracing dataset, using Layer Gradient \times Activation to compute importance scores until intermediate layers of the GPT2-XL model. **Bottom:** Visualization of contrastive attribution scores on a subset of layers (23 to 48) for some selected dataset examples. Plot labels show the contrastive pairs of false \rightarrow true answer used as attribution targets.

Pipeline for modeling causal beliefs from natural language

J Hunter Priniski

Department of Psychology
UCLA
priniski@ucla.edu

Ishaan Verma

Information Sciences Institute
USC
iverma@usc.edu

Fred Morstatter

Information Sciences Institute
USC
fredmors@isi.edu

Abstract

Causal reasoning is a core cognitive function and is central to how people learn and update their beliefs. Causal information is also central to how people represent and use language. Natural Language Processing algorithms that detect people’s causal representations can illuminate the considerations shaping their beliefs and reasoning. We present a causal language analysis pipeline that leverages a Large Language Model to identify causal claims in natural language documents, and aggregates claims across a corpus to produce a causal claim network. The pipeline then applies a clustering algorithm that groups causal claims according to their semantic topics. We demonstrate the pipeline by modeling causal belief systems surrounding the Covid-19 vaccine from tweets.

1 Introduction

Causal information facilitates learning (Holyoak and Cheng, 2011; Waldmann, 2007, 2017), and is crucial to how humans use and represent language (Mackie, 1980; Wolff et al., 2005; Lupyán, 2016). Causal relations are also ubiquitous in higher-level reasoning: they underlie our rich and flexible categories (Gelman and Legare, 2011), shape our explanatory preferences (Lombrozo and Vasilyeva, 2017), and structure our memories of events (Bower and Morrow, 1990).

Beliefs about causal relations can also have pernicious outcomes. For example, beliefs that vaccines cause autism are central to antivaccination attitudes (Horne et al., 2015; Powell et al., 2022), and the belief that liberal politicians have causal influence over the outcome of climate science research motivates climate change denialism (Cook and Lewandowsky, 2016). Because misinformation in online environments can spread rapidly to encourage these attitudes (Priniski et al., 2021; Priniski and Holyoak, 2022), new data science methods are necessary to combat these trends. However, data

science algorithms generally struggle to advance a rigorous scientific understanding of psychological processes, as they provide correlational evidence that does not isolate cognitive mechanisms. Methodologists should aim to develop Natural Language Processing (NLP) algorithms that produce cognitively plausible data representations that researchers can utilize to guide explanatory understanding and motivate future interventions.

Because causal relations are the backbone of most higher-level reasoning processes in humans and are central to how we use language, developing systems that can isolate people’s causal representations from language data is a natural place to start. However, NLP has historically struggled to identify instances of *psychological causality* (what a speaker *thinks* causes what) (Dunietz et al., 2017). This is because the variety of ways people communicate causality is immense (Talmy, 2000, 2011; Wolff, 2007), with most causal information latent in language inputs (Khoo et al., 2002; Blanco et al., 2008). Previously, methods that relied on hand labeling causal constructions to relate linguistic features to components of causal relations were extremely brittle and struggled to generalize to out-of-sample data (Yang et al., 2022). However, Large Language Models may help overcome this shortcoming as these models utilize rich semantic representations of language and sub-word tokenization that can help them identify instances of causal language not expressed in training (Devlin et al., 2018; Liu et al., 2019; Dunietz et al., 2017).

In addition to simply identifying instances of causal language, data representations should account for the breadth of people’s conceptual systems in which a causal claim is made. For example, causal beliefs are not held in isolation; instead, people have rich interlocking belief systems that span multiple topics that shape the integration of evidence (Quine and Ullian, 1978; Priniski and Holyoak, 2022; Gelman and Legare, 2011). Previ-

ous methods for producing representations of people’s belief systems rely on experiments and are slow to develop and may not generalize outside the lab (Powell, 2023; Powell et al., 2022). Because it is important to understand the full context of people’s belief systems to reliably predict how people will interpret evidence and make decisions, tools must be designed that can identify the vast web of beliefs that people use to interpret information in the wild. NLP tools can take advantage of the proliferation of social data online to build these representations (Goldstone and Lupyan, 2016).

To this end, we introduce a pipeline based on the Large Language Model, RoBERTa-XL (Liu et al., 2019), that detects causal claims in text documents, and aggregates claims made across a corpus to produce a network of interlocking causal claims, or a *causal claim network*¹. Causal claim networks can be used to approximate the beliefs and causal relations composing people’s conceptual understanding of the entities and events discussed in a corpus. To guide future research, we host a pipeline that produces interactive visualizations of people’s causal belief networks. We demonstrate this software by building causal belief systems surrounding Covid-19 vaccines from tweets.

2 How to build causal claim networks using our pipeline

The pipeline for extracting causal claim networks follows three main steps (see Figure 1). First, text documents are fed to a Large Language Model, a RoBERTa-XL transformer model (Liu et al., 2019), trained to extract causal claims made in a document (sentence to a paragraph in length). Second, the entities that compose causal claims are clustered according to their embeddings, clusters proxy *causal topics* (Grootendorst, 2022). Third, claims made across the corpus are coreferenced and strung together to make a network of cause and effect claims to be displayed in an interactive visualization.

We will now describe how a user could use our pipeline to build a causal claim network to visualize the causal claims made in a corpus of text documents. As shown in Figure 2, this follows two steps. First, a user uploads a .csv file containing the documents they wish to analyze. Documents should be a sentence to a paragraph in length and

¹We host the pipeline at the following link: <https://mindsgpu02.isi.edu:5020>. The code is available on GitHub: <https://github.com/ishaanverma/causal-claims-pipeline>.

can range from tweets, journal entries, or news headlines. Next, the user selects which column in the dataframe contains the texts to be analyzed. A user can also specify if they want the pipeline to preprocess the documents and cluster the entities. It is worth noting that entity clustering works best when there are an abundance of causal claims about semantically distinct entities. If a user chooses to cluster claims and the pipeline does not produce an output, it does not mean that there are no causal claims present, but rather that there are no clear semantic clusters. In these cases, the users should deselect ‘Cluster entities’ and rerun the pipeline.

As seen in Figure 2, we analyze a data set of tweets about the Covid-19 vaccine with the file name *covid_tweets.csv*, and the column containing the tweet texts is titled *tweets*. We provide access to this dataset on the tool interface, which can be downloaded to replicate this tutorial.

Once the document file is uploaded and the user presses submit, the job is queued and causal claims will be extracted. As shown in Figure 3, a job status window will be populated and the user will be updated on the degree of completion. As a rough reference, extracting causal claims from about 6000 tweets takes about a minute to complete once the job begins.

Once the job is completed, the screen will be populated with the causal claim network, like the one in Figure 4. There are a few things worth highlighting here. First, each edge represents a single extracted causal claim in the corpus, and nodes are colored by their causal cluster, or topic (see Figure 5). Clusters proxy topics in the data set and can be interpreted as central *causal topics* in the data set. In the next section, we describe how we calculate clusters.

The causal claim network produced by the pipeline is interactive. A user can click on an edge to see the document and extracted causal claim that constitutes that edge (see Figure 6). Furthermore, as shown in Figure 7, a user can simplify the network by selecting to collapse the edges between the nodes. The edge thickness is proportional to the number of documents between those two clusters. To facilitate downstream analysis of causal claims (e.g., by analyzing sentiment or stance of causal claims), a user can download the edge list that produced the network as a .csv file. Columns in this .csv file include: cause word span, cause cluster, effect word span, effect cluster, text, and

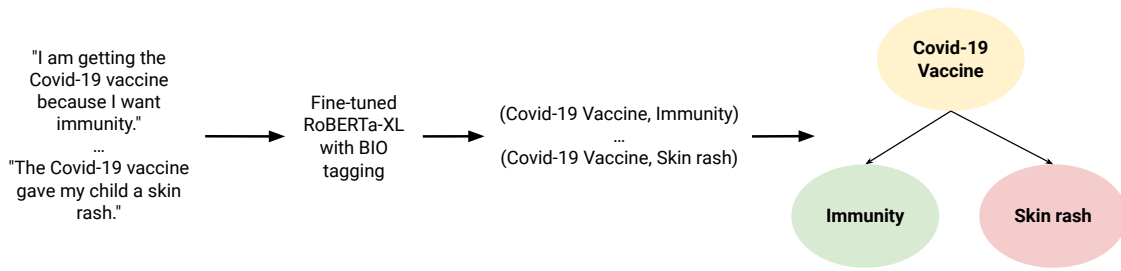


Figure 1: High-level schematic describing how text documents become a causal claim network. Raw text documents are first fed to a RoBERTa-XL transformer fine-tuned on causal language, which follows a BIO tagging scheme (see Section 3.1) to return a list of tuples encoding the expressed cause and effect relationships. These tuples are co-referenced across the corpus to produce a causal claim network.

The screenshot shows a web interface with two main steps. Step 1, "Upload a CSV", includes a "Choose a file..." button and a text input field containing "covid_data.csv", with a "Submit" button below. Step 2, "Select a column", features a dropdown menu set to "tweet", two checked checkboxes labeled "Pre-process text (remove URLs, email, extra whitespace)" and "Cluster entities", and a "Submit" button. A note below the checkboxes states: "Large CSV files are automatically truncated to the first 10,000 rows".

Figure 2: Uploading text data for causal claim analysis follows two steps. First select the .csv file you wish to analyze, then select which column in the dataframe contains the text documents to be analyzed.

The screenshot shows a "Job Status" window with a dark header. Below the header, the following text is displayed: "Job id: 1eebce8a-03c9-4708-9c05-300723c96562", "Status: Running causal model...", and "1314 / 6092". A blue progress bar is shown below the text, indicating the job's progress.

Figure 3: Job queue status window.

document id.

The pipeline allows users to specify different parameters for the model (see Figure 8). While the pipeline uncovers clusters automatically, users can specify the number of clusters to uncover in the corpus (this will equate to the number of nodes in the causal claim network). Users can also specify the N_gram range to be used during preprocessing and can specify the number of top words used to describe each causal cluster/topic.

3 What's happening under the hood

In this section, we describe how the pipeline builds causal claim networks. This follows three steps: (1) causal claims are extracted using a RoBERTa-XL transformer model that identifies which words belong to cause and effect events (Li et al., 2021), (2) claims are clustered based on their semantic topics, and (3) a causal claim network is built by combining the claims stated in the corpus.

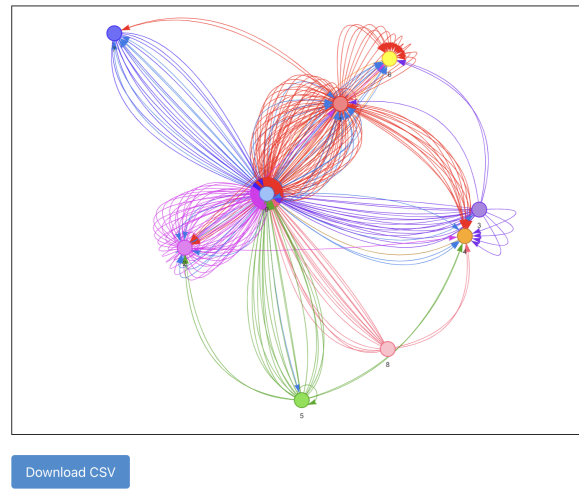


Figure 4: A causal claim network built from tweets about the Covid-19 vaccine. Individual nodes denote broad causal topics (i.e., clusters of cause and effect word spans based on their semantic embeddings), and edges signify a document containing a causal claim linking entities belonging to those two clusters.

3.1 Step 1: Extracting causal claims

Documents are first fed to a RoBERTa-XL transformer network fine-tuned to identify the cause and effect pairs of nominals in natural language documents (Hendrickx et al., 2010; Li et al., 2021). The training set consists of 4,450 sentences and contains 1,570 causal relations, and the test set consists of 804 sentences with 296 causal relations. Following the SCITE architecture (Li et al., 2021), we set up training as a token classification task, where we utilize the before-inside-outside (BIO) labeling scheme to identify which words belong to a cause-span, effect-span, or embedded-causality-span (tokens belonging to a causal event in the middle of a causal chain). As seen in Table 1, RoBERTa-XL has a higher performance than previous state-of-the-art models on this task (Li et al., 2021), and performs better than the smaller BERT



Figure 5: Causal clusters, or causal topics, are shown to the right of the produced causal claim network. Each topic consists of a set of keywords that describes the cluster.

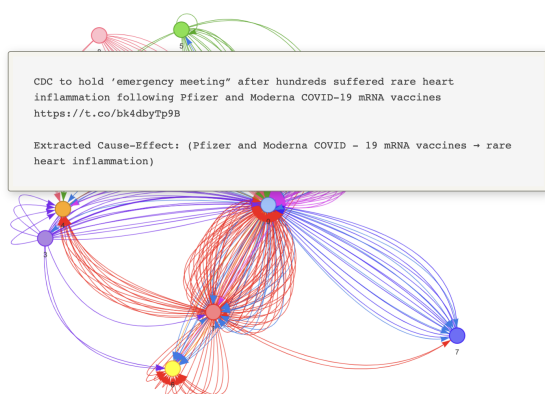
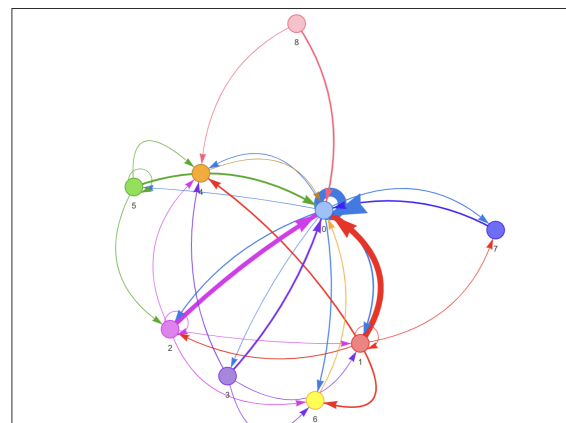


Figure 6: Hovering over an edge in the causal claim network displays the document and extracted causal claim that constitutes that edge. The document is shown at the top of the box, and the extracted cause claim is at the bottom.



Download CSV

Visualization Options

Merge Edges

Figure 7: Causal claim network with merged edges, where edge weights equates to the number of documents linking two clusters. Merging edges is useful to quickly assess degree of linkage between causal clusters (nodes) in the network.

Cluster Options

Number of Clusters

After training the topic model, the number of topics that will be reduced. For example, if the topic model results in 100 topics but you have set `nr_topics` to 20 then the topic model will try to reduce the number of topics from 100 to 20.

Setting this value to 0 will automatically reduce topics using HDBSCAN. Setting this value to -1 will not perform topic reduction. Default value is 0.

0

N-gram range

It relates to the number of words you want in your topic representation. For example, "New" and "York" are two separate words but are often used as "New York" which represents an n-gram of 2. Thus, the `n_gram_range` should be set to (1, 2) if you want "New York" in your topic representation. Default value is (1, 2).

1 2

Top N words

Refers to the number of words per topic that you want to be extracted. Default value is 10.

10

Submit

Figure 8: A user can specify parameters when running the pipeline to engage with exploratory data analysis. Users can specify the number of clusters, the n-gram range used during processing, and set the number of words to describe each topic.

	Precision	Recall	F1
SCITE	0.833	0.858	0.845
BERT	0.824	0.858	0.841
RoBERTa-XL	0.883	0.865	0.874

Table 1: Model performance on the causal relation identification task (Hendrickx et al., 2010). The RoBERTa-XL model demonstrates increased performance over the smaller transformer BERT and previously reported state-of-the-art implementations (Li et al., 2021)

transformer (Devlin et al., 2018). We therefore used the RoBERTa-XL transformer in our pipeline.

Once the sentences have been tagged using the causality tagging scheme, we run the tag2triplet algorithm proposed by Li et al., 2021 to extract the cause-effect tuples from the tagged sequence. The algorithm operates by first identifying the in-degree and out-degree of causality in the tagged sequence. Here, if the entity is labeled as a “cause”, then the out-degree is increased by 1; if the entity is labeled as an “effect” then the in-degree is incremented by 1; and if the entity is labeled as “embedded causality” then both the in-degree and out-degree are incremented by 1. The algorithm then tries to align the identified entities such that each entity that has an outgoing edge (i.e., the cause) is joined with the entity that has an incoming edge (i.e., the effect) while taking into consideration the distance between the entities in the document and whether they contain a coordinating conjunction.

3.2 Step 2: Finding causal topics by clustering embeddings

Clusters of causes and effects proxy topics in the causal claim network. We cluster the embeddings of the nodes by extending the tf-idf measure of the embeddings (Grootendorst, 2022). This method was originally developed to cluster BERT representations to uncover topics in a corpus, but we implemented the algorithm to cluster RoBERTa-XL embeddings. This allows users to assess latent structure in the causal claims expressed in a corpus and simplifies the resulting causal claim graph by mapping semantically similar claims to a common node.

3.3 Constructing a causal claim network

Extracted cause-effect tuples serve as directed edges in the causal claim network, which are strung together throughout the corpus to form a causal claim network. Nodes are the identified cause and

effect wordspans and the weighted edges encode the number of instances in the corpus where node i was said to cause node j . The edge direction encodes the direction of the causal relation and can be supplied with additional semantic content (e.g., relational vectors, sentiment).

4 Case study: Building a network of causal claims about the Covid-19 vaccine from tweets

4.1 Data set of tweets

To test this pipeline, we build a causal claim network using a set of 6000 tweets about the Covid-19 vaccine (Poddar et al., 2022). The original dataset was curated by subsetting a larger sample of tweets from before and after the release of the Covid-19 vaccines.

4.2 Pipeline results

The pipeline returns 408 extracted causal claims belonging to nine distinct clusters (see Figure 5). The clusters are, as expected, about the various Covid-19 vaccines and their anticipated consequences. By aggregating the keywords for each cluster, we can define the set of causal topics returned by the pipeline. More specifically, cluster 0 contains keywords related to *Death*; 1: *Oxford vaccines*, 2: *Covid-19 pandemic*; 3: *Pfizer vaccine*, 4: *Side-effects*, 5: *Pfizer shot*, 6: *Immunity and antibodies*, 7: *Coronavirus*, and 8: *Covid vaccine*. As shown in Table 2, we see that the clusters are about a range of topics with varying semantics and valence, which suggests that the pipeline can help us understand the breadth of considerations guiding Twitter discussions about the Covid-19 vaccine.

4.3 Secondary analyses of extracted causal claims

By analyzing the causal claims returned by the pipeline (which is also available for download as a .csv file), we can explore how these causal clusters are linked to one another. For example, as shown in 2, some of the clusters are more commonly composed of word spans denoting cause events, while others are more composed of word spans denoting effect events.

5 Related work

Although our approach is domain-general (documents do not need to belong to a single issue or topic for the pipeline to return a set of causal

Cluster	Topic	Causes	Effects
0	Death	154	299
1	Oxford vaccine	108	15
2	Pandemic	56	16
3	Pfizer vaccine	25	1
4	Side-effects	1	25
5	Pfzier shot	22	2
6	Immunity	4	29
7	Coronavirus	13	8
8	Covid vaccine	17	0

Table 2: Number of identified word spans per each causal cluster. The topic label is determined by assessing the top keywords in each causal cluster. Each cluster has a different distribution of cause and effect spans.

claims), we demonstrate the use of our pipeline modeling causal claims about the Covid-19 vaccine. Previous work has developed systems specifically designed to analyze claims about Covid-19. For example, Li et al. (2022) built a system specifically designed to monitor claims made about Covid-19. This system identifies claims and arguments made in the corpus, and sources additional Wikidata information to put the claims in a richer content.

Mining causal claim networks requires isolating causal claims which oftentimes constitute arguments expressed in a text document: a reasoner makes a causal claim when explaining a mechanism (Lombrozo and Vasilyeva, 2017) or argument. Claim detection is an active area of research (Palau and Moens, 2009; Goudas et al., 2014), as is the detection of the components of arguments in text (Sardianos et al., 2015). Because causal reasoning is central to the way people construct arguments (Abend et al., 2013), understanding how people posit causal claims can shed light on the types of arguments people will endorse related to that issue.

Most claim detection algorithms work on the level of single documents, but approaches such as those of Levy et al. 2014 propose corpus-wide claim detection. Our pipeline utilizes a mixing of the two: claims are detected within a document and then aggregated across the documents in the corpus to provide a corpus-level representation.

6 Conclusions and future work

Interactive data visualization is an effective way for people to make sense of complex data (Janvrin et al., 2014) and can be an effective tool to guide scientific thinking (Franconeri et al., 2021). Our

pipeline is designed to help researchers explore the causal claims expressed in a corpus through interactive exploration.

There are therefore many applications of this tool to the study of human reasoning and belief change, and future work will test the efficacy of these use cases. For example, researchers in cognitive science have worked on developing methods to measure people’s rich conceptual systems about vaccines (Powell, 2023). These methods often require the development of surveys that measure people’s attitudes toward a variety of related issues. Causal claim networks can give researchers a starting place to know what measures they should include in these surveys.

In future work, we will work on expanding the visualization tool to include features that allow for richer forms of interaction. For example, by allowing users to build subnetworks based on another data attribute (e.g., stance of the document, expressed sentiment), to allow for comparisons across networks. Related to this, future work will also develop quantitative measures of divergence across networks.

Acknowledgements

This work was funded in part by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under Contract No. W911NF-21-C-0002 and AFOSR MURI grant number FA9550-22-1-0380.

References

- Gabriel Abend, Caitlin Petre, and Michael Sauder. 2013. Styles of causal thought: An empirical investigation. *American Journal of Sociology*, 119(3):602–654.
- Eduardo Blanco, Nuria Castell, and Dan Moldovan. 2008. Causal relation extraction. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*.
- Gordon H Bower and Daniel G Morrow. 1990. Mental models in narrative comprehension. *Science*, 247(4938):44–48.
- John Cook and Stephan Lewandowsky. 2016. Rational irrationality: Modeling climate change belief polarization using bayesian networks. *Topics in Cognitive Science*, 8(1):160–179.
- 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*.

- Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. Automatically tagging constructions of causation and their slot-fillers. *Transactions of the Association for Computational Linguistics*, 5:117–133.
- Steven L Franconeri, Lace M Padilla, Priti Shah, Jeffrey M Zacks, and Jessica Hullman. 2021. The science of visual data communication: What works. *Psychological Science in the public interest*, 22(3):110–161.
- Susan A Gelman and Cristine H Legare. 2011. Concepts and folk theories. *Annual Review of Anthropology*, 40:379.
- Robert L Goldstone and Gary Lupyan. 2016. Discovering psychological principles by mining naturally occurring data sets. *Topics in Cognitive Science*, 8(3):548–568.
- Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument extraction from news, blogs, and social media. In *Artificial Intelligence: Methods and Applications: 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15-17, 2014. Proceedings 8*, pages 287–299. Springer.
- Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.
- Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. [SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals](#). In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.
- Keith J Holyoak and Patricia W Cheng. 2011. Causal learning and inference as a rational process: The new synthesis. *Annual Review of Psychology*, 62:135–163.
- Zachary Horne, Derek Powell, John E Hummel, and Keith J Holyoak. 2015. Countering antivaccination attitudes. *Proceedings of the National Academy of Sciences*, 112(33):10321–10324.
- Diane J Janvrin, Robyn L Raschke, and William N Dilla. 2014. Making sense of complex data using interactive data visualization. *Journal of Accounting Education*, 32(4):31–48.
- Christopher Khoo, Syin Chan, and Yun Niu. 2002. The many facets of the cause-effect relation. In *The Semantics of Relationships*, pages 51–70. Springer.
- Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. [Context dependent claim detection](#). In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Manling Li, Revanth Gangi Reddy, Ziqi Wang, Yi-Shyuan Chiang, Tuan Lai, Pengfei Yu, Zixuan Zhang, and Heng Ji. 2022. Covid-19 claim radar: A structured claim extraction and tracking system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 135–144.
- Zhaoning Li, Qi Li, Xiaotian Zou, and Jiangtao Ren. 2021. Causality extraction based on self-attentive bilstm-crf with transferred embeddings. *Neurocomputing*, 423:207–219.
- 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*.
- Tania Lombrozo and Nadya Vasilyeva. 2017. Causal explanation. *The Oxford Handbook of Causal Reasoning*, page 415.
- Gary Lupyan. 2016. The centrality of language in human cognition. *Language Learning*, 66(3):516–553.
- John Leslie Mackie. 1980. *The cement of the universe: A study of causation*. Clarendon Press.
- Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107.
- Soham Poddar, Mainack Mondal, Janardan Misra, Niloy Ganguly, and Saptarshi Ghosh. 2022. Winds of change: Impact of covid-19 on vaccine-related opinions of twitter users. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 16, pages 782–793.
- Derek Powell. 2023. Comparing probabilistic accounts of probability judgments. *Computational Brain & Behavior*, pages 1–18.
- Derek Powell, Kara Weisman, and Ellen Markman. 2022. Modeling and leveraging intuitive theories to improve vaccine attitudes. *PsyArXiv*.
- J Hunter Priniski and Keith J Holyoak. 2022. A darkening spring: How preexisting distrust shaped covid-19 skepticism. *PloS one*, 17(1):e0263191.
- J Hunter Priniski, Mason McClay, and Keith J Holyoak. 2021. Rise of qanon: A mental model of good and evil stews in an echochamber. *arXiv preprint arXiv:2105.04632*.
- Willard Van Orman Quine and Joseph Silbert Ullian. 1978. *The Web of Belief*, volume 2. Random House: New York.

- Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66.
- Leonard Talmy. 2000. *Toward a cognitive semantics*, volume 2. MIT press.
- Leonard Talmy. 2011. 27. *Cognitive Semantics: An overview*, pages 622–642. De Gruyter Mouton, Berlin, Boston.
- Michael R Waldmann. 2007. Combining versus analyzing multiple causes: How domain assumptions and task context affect integration rules. *Cognitive science*, 31(2):233–256.
- Michael R Waldmann. 2017. Causal reasoning: An introduction. *The Oxford Handbook of Causal Reasoning*, pages 1–9.
- Phillip Wolff. 2007. Representing causation. *Journal of Experimental Psychology: General*, 136(1):82.
- Phillip Wolff, Bianca Klettke, Tatyana Ventura, and Grace Song. 2005. Expressing causation in english and other languages. In *Categorization inside and outside the laboratory: Essays in honor of Douglas L. Medin*, pages 29–48. American Psychological Association.
- Jie Yang, Soyeon Caren Han, and Josiah Poon. 2022. A survey on extraction of causal relations from natural language text. *Knowledge and Information Systems*, pages 1–26.

TABGENIE: A Toolkit for Table-to-Text Generation

Zdeněk Kasner¹ Ekaterina Garanina^{1,2} Ondřej Plátek¹ Ondřej Dušek¹

¹Charles University, Czechia

²University of Groningen, The Netherlands

{kasner, oplatek, odusek}@ufal.mff.cuni.cz

e.garanina@student.rug.nl

Abstract

Heterogeneity of data-to-text generation datasets limits the research on data-to-text generation systems. We present TABGENIE – a toolkit which enables researchers to explore, preprocess, and analyze a variety of data-to-text generation datasets through the unified framework of *table-to-text generation*. In TABGENIE, all inputs are represented as tables with associated metadata. The tables can be explored through a web interface, which also provides an interactive mode for debugging table-to-text generation, facilitates side-by-side comparison of generated system outputs, and allows easy exports for manual analysis. Furthermore, TABGENIE is equipped with command line processing tools and Python bindings for unified dataset loading and processing. We release TABGENIE as a PyPI package¹ and provide its open-source code and a live demo at <https://github.com/kasnerz/tabgenie>.

1 Introduction

Building and evaluating data-to-text (D2T) generation systems (Gatt and Krahmer, 2018; Sharma et al., 2022) requires understanding the data and observing system behavior. It is, however, not trivial to interact with the large volume of D2T generation datasets that have emerged in the last years (see Table 1). Although research on D2T generation benefits from platforms providing unified interfaces, such as HuggingFace Datasets (Lhoest et al., 2021) or the GEM benchmark (Gehrmann et al., 2021), these platforms still leave the majority of the data processing load on the user.

A key component missing from current D2T tools is the possibility to visualize the input data and generated outputs. Visualization plays an important role in examining and evaluating scientific data (Kehrer and Hauser, 2013) and can help D2T generation researchers to make more informed de-

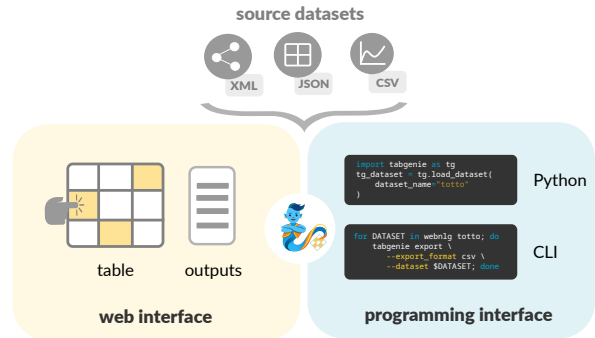


Figure 1: TABGENIE provides a way to handle various data-to-text generation datasets through a unified web and programming interface. The *web interface* enables interactive exploration and analysis of datasets and model outputs, while the *programming interface* provides unified data loaders and structures.

sign choices. A suitable interface can also encourage researchers to step away from unreliable automatic metrics (Gehrmann et al., 2022) and focus on manual error analysis (van Miltenburg et al., 2021, 2023).

Along with that, demands for a *unified input data format* have recently been raised with multi-task training for large language models (LLMs) (Sanh et al., 2022; Scao et al., 2022; Ouyang et al., 2022, *inter alia*). Some works have used simple data linearization techniques for converting structured data to a textual format, in order to align it with the format used for other tasks (Xie et al., 2022; Tang et al., 2022). However, linearizations are using custom preprocessing code, leading to discrepancies between individual works.

In this paper, we present TABGENIE – a multi-purpose toolkit for interacting with D2T generation datasets and systems designed to fill these gaps. On a high level, the toolkit consists of (a) an interactive web interface, (b) a set of command-line processing tools, and (c) a set of Python bindings (see Figure 1).

The cornerstone of TABGENIE is a **unified data**

¹<https://pypi.org/project/tabgenie/>

Dataset	Source	Data Type	Number of examples			License
			train	dev	test	
CACAPO	van der Lee et al. (2020)	Key-value	15,290	1,831	3,028	CC BY
DART†	Nan et al. (2021)	Graph	62,659	2,768	5,097	MIT
E2E†	Dusek et al. (2019)	Key-value	33,525	1,484	1,847	CC BY-SA
EventNarrative	Colas et al. (2021)	Graph	179,544	22,442	22,442	CC BY
HiTab	Cheng et al. (2022)	Table w/hl	7,417	1,671	1,584	C-UDA
Chart-To-Text	Kantharaj et al. (2022)	Chart	24,368	5,221	5,222	GNU GPL
Logic2Text	Chen et al. (2020b)	Table w/hl + Logic	8,566	1,095	1,092	MIT
LogicNLG	Chen et al. (2020a)	Table	28,450	4,260	4,305	MIT
NumericNLG	Suadaa et al. (2021)	Table	1,084	136	135	CC BY-SA
SciGen	Moosavi et al. (2021)	Table	13,607	3,452	492	CC BY-NC-SA
SportSett:Basketball†	Thomson et al. (2020)	Table	3,690	1,230	1,230	MIT
ToTTo†	Parikh et al. (2020)	Table w/hl	121,153	7,700	7,700	CC BY-SA
WebNLG†	Ferreira et al. (2020)	Graph	35,425	1,666	1,778	CC BY-NC
WikiBio†	Lebret et al. (2016)	Key-value	582,659	72,831	72,831	CC BY-SA
WikiSQL†	Zhong et al. (2017)	Table + SQL	56,355	8,421	15,878	BSD
WikiTableText	Bao et al. (2018)	Key-value	10,000	1,318	2,000	CC BY

Table 1: The list of datasets included in TABGENIE. Glossary of data types: *Key-value*: key-value pairs, *Graph*: subject-predicate-object triples, *Table*: tabular data (*w/hl*: with highlighted cells), *Chart*: chart data, *Logic / SQL*: strings with logical expressions / SQL queries. The datasets marked with † were already present on Huggingface Datasets. We uploaded the rest of the datasets to our namespace: <https://huggingface.co/kasnerz>.

representation. Each input represented is as a matrix of m columns and n rows consisting of individual cells accompanied with metadata (see §2). Building upon this representation, TABGENIE then provides multiple features for unified workflows with table-to-text datasets, including:

1. visualizing individual dataset examples in the tabular format (§3.1),
2. interacting with table-to-text generation systems in real-time (§3.2),
3. comparing generated system outputs (§3.2),
4. loading and preprocessing data for downstream tasks (§4.1),
5. exporting examples and generating spreadsheets for manual error analysis (§4.2).

In §6, we present examples of practical use-cases of TABGENIE in D2T generation research.

2 Data

We currently include 16 datasets listed in Table 1 in TABGENIE, covering many subtasks of D2T generation. All the datasets are available under a permissive open-source license.

2.1 Data Format

The inputs in D2T generation datasets may not consist only of tables, but also of e.g. graphs or key-value pairs. However, we noticed that in many

cases, converting these formats to tables requires only minimal changes to the data structure while allowing a unified data representation and visualization. This conversion narrows down the task of D2T generation as the task of generating description for a tabular data, i.e. table-to-text generation (Parikh et al., 2020; Liu et al., 2022; Gong et al., 2020).

In our definition, a *table* is a two-dimensional matrix with m columns and n rows, which together define a grid of $m \times n$ cells. Each cell contains a (possibly empty) text string. A continuous sequence of cells $\{c_i, \dots, c_{i+k}\}$ from the same row or column may be merged, in which case the values of $\{c_{i+1}, \dots, c_{i+k}\}$ are linked to the value of c_i . A cell may be optionally marked as a *heading*, which is represented as an additional property of the cell.² To better accommodate the format of datasets such as ToTTo (Parikh et al., 2020) or HiTab (Cheng et al., 2022), we also allow individual cells to be *highlighted*. Highlighted cells are assumed to be preselected for generating the output description.

The tables may be accompanied with an additional set of properties (see Figure 2) – an example of such a property is a “*title*” of the table in WikiBio (Lebret et al., 2016) or a “*category*” in WebNLG (Gardent et al., 2017). We represent prop-

²The headings are typically located in the first row or column, but may also span multiple rows or columns and may not be adjacent.

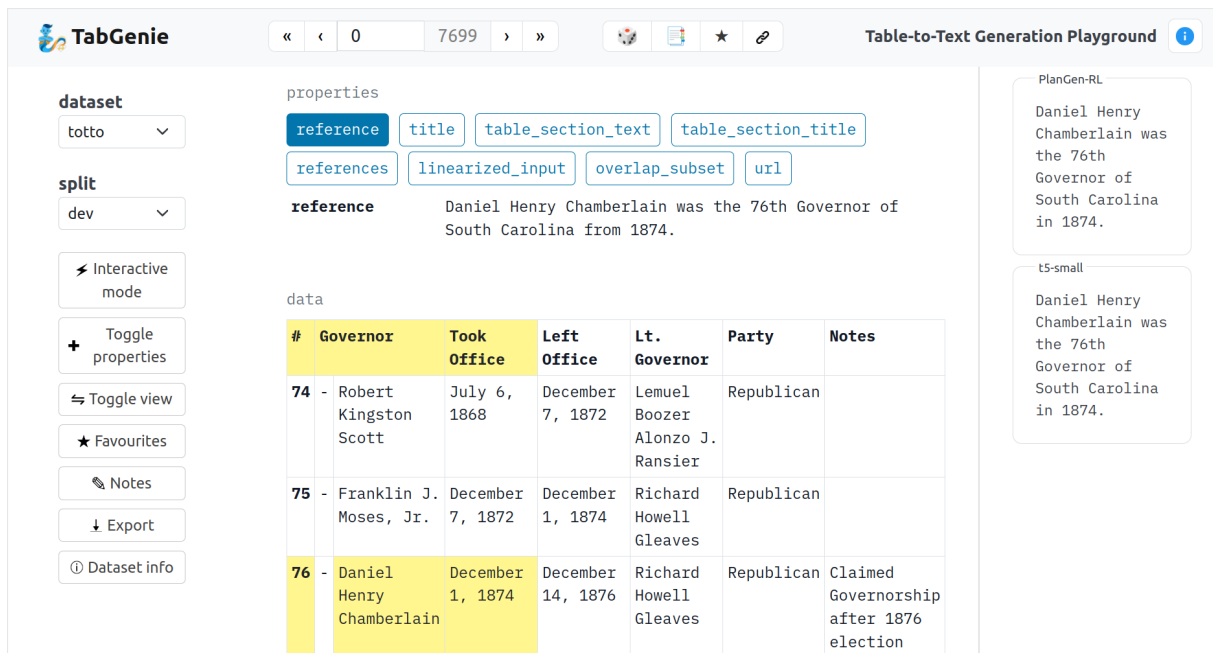


Figure 2: The web interface of TABGENIE. The **left panel** and the **navigation bar** contains user controls; the **center panel** shows table properties and table content; the **right panel** contains system outputs.

erties as key-value pairs alongside the table. The properties may be used for generating the table description.

2.2 Data Transformation

We aim to present the data as true to the original format as possible and only make some minor changes for datasets which do not immediately adhere to the tabular format:

- For graph-to-text datasets, we format each triple as a row, using three columns labeled *subject*, *predicate*, and *object*.
- For key-value datasets, we use two columns with keys in the first column as row headings.
- For SportSett:Basketball (Thomson et al., 2020), we merge the *box score* and *line score* tables and add appropriate headings where necessary.

Moreover, for ToTTo (Parikh et al., 2020), we also provide our custom, improved header cells highlighting (details are given in Appendix A).

2.3 Data Loading

To ease the data distribution, we load all the datasets using the Huggingface datasets package (Lhoest et al., 2021), which comes equipped with a data downloader. Out of 16 datasets we are using, 7 were already available in Huggingface datasets,

either through the GEM benchmark (Gehrmann et al., 2021) or other sources. We publicly added the 9 remaining datasets (see Table 1).

TABGENIE also supports adding custom data loaders. Creating a data loader consists of simple sub-classing the data loader class and overriding a single method for processing individual entries, allowing anyone to add their custom dataset.

3 Web Interface

TABGENIE offers a user-friendly way to interact with table-to-text generation datasets through the *web interface*. The interface can be rendered using a local server (cf. §4.2) and can be viewed in any modern web browser. The interface features a simple, single-page layout, which contains a navigation bar and three panels containing user controls, input data, and system outputs (see Figure 2). Although the interface primarily aims at researchers, it can be also used by non-expert users.

3.1 Content Exploration

TABGENIE renders input data as HTML tables. This approach provides superior visualizations to existing data viewers, especially in the case of large and hierarchical tables.³

³Compare, e.g., with the ToTTo dataset in Huggingface Datasets for which the table is provided in a single field called “table”: <https://huggingface.co/datasets/totto>

In the web interface, users can navigate through individual examples in the dataset sequentially, access an example using its index, or go to a random example. The users can add notes to examples and mark examples as favorites for accessing them later. The interface also shows the information about the dataset (such as its description, version, homepage, and license) and provides an option to export the individual examples (see §4.2).

3.2 Interactive Mode

TABGENIE offers an *interactive mode* for generating an output for a particular example on-the-fly. The user can highlight different cells, edit cell contents, and edit parameters of the downstream processor. For example, the user can prompt a LLM for table-to-text generation and observe how it behaves while changing the prompt.

The contents of a table are processed by a processing *pipeline*. This pipeline takes table contents and properties as input, processes them with a sequence of modules, and outputs HTML code. The modules are custom Python programs which may be re-used across the pipelines.

TABGENIE currently provides two basic pipelines: (1) calling a generative language model through an API with a custom prompt, and (2) generating graph visualizations of RDF triples. We describe a case-study for the model API pipeline in §6.2. Users can easily add custom pipelines by following the instructions in the project repository.

3.3 Pre-generated Outputs

In addition to interactive generation, TABGENIE allows users to visualize static pre-generated outputs. These are loaded in the JSONL⁴ format from a specified directory and displayed similarly to model-generated outputs from the interactive mode. Multiple outputs can be displayed alongside a specific example, allowing to compare outputs from multiple systems.

4 Developer Tools

TABGENIE also provides a developer-friendly interface: Python bindings (§4.1) and a command-line interface (§4.2). Both of these interfaces aim to simplify dataset preprocessing in downstream tasks. The key benefit of using TABGENIE is that it provides streamlined access to data in a consistent format, removing the need for dataset-specific code

⁴<https://jsonlines.org>

for extracting information such as table properties, references, or individual cell values.

4.1 Python Bindings

TABGENIE can be integrated in other Python codebases to replace custom preprocessing code. With a *single unified interface* for all the datasets, the TABGENIE wrapper class allows to:

- load a dataset from the Huggingface Datasets or from a local folder,
- access individual table cells and their properties,
- linearize tables using pre-defined or custom functions,
- prepare the Huggingface Dataset objects for downstream processing.

TABGENIE can be installed as a Python package, making the integration simple and intuitive. See §6.1 for an example usage of the TABGENIE Python interface.

4.2 Command-line Tools

TABGENIE supports several basic commands via command line.

Run The `tabgenie run` command launches the local web server, mimicking the behavior of `flask run`. Example usage:

```
tabgenie run --port=8890 --host="0.0.0.0"
```

Export The `tabgenie export` command enables batch exporting of the dataset. The supported formats are `xlsx`, `html`, `json`, `txt`, and `csv`. Except for `csv`, table properties can be exported along with the table content. Example usage:

```
tabgenie export --dataset "webnlg" \  
--split "dev" \  
--out_dir "export/datasets/webnlg" \  
--export_format "xlsx"
```

Export can also be done in the web interface.

Spreadsheet For error analysis, it is common to select N random examples from the dataset along with the system outputs and manually annotate them with error categories (see §6.3). The `tabgenie sheet` command generates a suitable spreadsheet for this procedure. Example usage:

```
tabgenie sheet --dataset "webnlg" \
--split "dev" \
--in_file "out-t5-base.jsonl" \
--out_file "analysis_webnlg.xlsx" \
--count 50
```

5 Implementation

TABGENIE runs with Python ≥ 3.8 and requires only a few basic packages as dependencies. It can be installed as a stand-alone Python module from PyPI (`pip install tabgenie`) or from the project repository.

Backend The web server is based on Flask,⁵ a popular lightweight Python-based web framework. The server runs locally and can be configured with a YAML⁶ configuration file. On startup, the server loads the data using the datasets⁷ package. To render web pages, the server uses the tinyhtml⁸ package and the Jinja⁹ templating language. We provide details on the computational and memory requirements in Appendix D.

Frontend The web frontend is built on HTML5, CSS, Bootstrap,¹⁰ JavaScript, and jQuery.¹¹ We additionally use the D3.js¹² library for visualizing the structure of data in graph-to-text datasets. To keep the project simple, we do not use any other major external libraries.

6 Case Studies

In this section, we outline several recipes for using TABGENIE in D2T generation research. The instructions and code samples for these tasks are available in the project repository.

6.1 Table-To-Text Generation

Application Finetuning a sequence-to-sequence language model for table-to-text generation in PyTorch (Paszke et al., 2019) using the Huggingface Transformers (Wolf et al., 2020) framework.

Process In a typical finetuning procedure using these frameworks, the user needs to prepare a Dataset object with tokenized input and output

sequences. Using TABGENIE, preprocessing a specific dataset is simplified to the following:

```
from transformers import AutoTokenizer
import tabgenie as tg

# instantiate a tokenizer
tokenizer = AutoTokenizer.from_pretrained(...)

# load the dataset
tg_dataset = tg.load_dataset(
    dataset_name="totto"
)

# preprocess the dataset
hf_dataset = tg_dataset.get_hf_dataset(
    split="train",
    tokenizer=tokenizer
)
```

The function `get_hf_dataset()` linearizes the tables (the users may optionally provide their custom linearization function) and tokenizes the inputs and references.

For training a single model on multiple datasets in a multi-task learning setting (Xie et al., 2022), the user may preprocess each dataset individually, prepending a dataset-specific task description to each example. The datasets may then be combined using the methods provided by the datasets package.

Demonstration For running the baselines, we provide an example script, which can be applied to any TABGENIE dataset and pre-trained sequence-to-sequence model from the transformers library. For multi-task learning, we provide an example of joint training on several datasets with custom linearization functions. We run the example scripts for several datasets and display the resulting generations in the application demo. Details on the fine-tuned models can be found in Appendix B.

6.2 Interactive Prompting

Application Observing the impact of various inputs on the outputs of a LLM prompted for table-to-text generation.

Process The user customizes the provided `model_api` pipeline to communicate with a LLM through an API. The API can communicate either with an external model (using e.g. OpenAI API¹³), or with a model running locally (using libraries such as FastAPI¹⁴). The user then interacts with the model through TABGENIE web interface, modifying the prompts, highlighted cells, and table content (see §3.2).

⁵<https://pypi.org/project/Flask/>

⁶<https://yaml.org>

⁷<https://pypi.org/project/datasets/>

⁸<https://pypi.org/project/tinyhtml/>

⁹<https://jinja.palletsprojects.com/>

¹⁰<https://getbootstrap.com/>

¹¹<https://jquery.com>

¹²<https://d3js.org>

¹³<https://openai.com/api/>

¹⁴<https://fastapi.tiangolo.com>

Demonstration We provide an interactive access to the instruction-tuned Tk-Instruct LLM (Wang et al., 2022) in the project live demo. The user can use the full range of possibilities included in the interactive mode, including customizing the prompt and the input data.¹⁵ The interface is shown in Appendix C.

6.3 Error Analysis

Application Annotating error categories in the outputs from a table-to-text generation model.

Process The user generates the system outputs (see §6.1) and saves the outputs for a particular dataset split in a JSONL format. Through the command-line interface, the user will then generate a XLSX file which can be imported in any suitable office software and distributed to annotators for performing error analysis.

Demonstration We provide instructions for generating the spreadsheet in the project documentation. See Appendix C for a preview of the spreadsheet format.

7 Related Work

7.1 Data Loading and Processing

As noted throughout the work, Huggingface Datasets (Lhoest et al., 2021) is a package commonly used for data loading and preprocessing. TABGENIE serves as a wrapper on top of this package, providing additional abstractions and better data visualization for D2T generation datasets.

DataLab (Xiao et al., 2022) is another platform for working with NLP datasets. Similarly to Huggingface Datasets, this platform has much broader focus than our package. Besides data access, it offers fine-grained data analysis and data manipulation tools. However, it has limited capabilities of visualizing the input data or interactive generation and at present, it does not cover the majority of datasets available in TABGENIE.

PromptSource (Bach et al., 2022) is a framework for constructing prompts for generative language models using the Jinja templating language. It can be used both for developing new prompts and for using the prompts in downstream applications.

¹⁵Note that using the model for the task of table-to-text generation is experimental and may not produce optimal outputs. The model should also not be used outside of demonstration purposes due to our limited computational resources.

Several tools have been developed for comparing outputs of language generation systems (notably for machine translation) such as CompareMT (Neubig et al., 2019) or Appraise (Federmann, 2018), but the tools do not visualize the structured data.

7.2 Interactive D2T Generation

Platforms for interactive D2T generation have been primarily limited to commercial platforms, such as Arria,¹⁶ Automated Insights,¹⁷ or Tableau Software¹⁸ (formerly Narrative Science). These platforms focus on proprietary solutions for generating business insights and do not provide an interface for research datasets. Dou et al. (2018) present Data2Text Studio, a set of developer tools for building custom D2T generation systems, but their software package currently does not seem to be publicly available.

7.3 Table-To-Text Generation

Although pre-trained sequence-to-sequence models have been found to be effective for D2T generation (Kale and Rastogi, 2020; Xie et al., 2022), they have difficulties with handling the input structure, generation diversity, and logical reasoning. Multiple works have tried to address these issues. For a comprehensive review of the field, we point the interested reader to the recent survey of Sharma et al. (2022).

8 Conclusion

We presented TABGENIE, a multifunctional software package for table-to-text generation. TABGENIE bridges several gaps including visualizing input data, unified data access, and interactive table-to-text generation. As such, TABGENIE provides a comprehensive set of tools poised to accelerate progress in the field of D2T generation.

Limitations

For some D2T generation inputs, the tabular structure may be inappropriate. This involves hierarchical tree-based structures, bag-of-words, or multimodal inputs (Balakrishnan et al., 2019; Lin et al., 2019; Krishna et al., 2017). Due to deployment issues, TABGENIE also does not include large synthetic datasets (Agarwal et al., 2021; Jin et al.,

¹⁶<https://www.arria.com>

¹⁷<https://automatedinsights.com>

¹⁸<https://www.tableau.com>

2020). TABGENIE is currently in early development stages, which is why it primarily targets the research community.

Ethical Impact

The table-to-text generation datasets may contain various biases or factually incorrect outputs, which may be further reproduced by the table-to-text generation models. Although our software package is designed to help to examine and eliminate the biases and errors, we cannot guarantee the correctness of the processed outputs.

As TABGENIE is an open-source software package with a permissive license, we do not control its downstream applications. We advocate using it for responsible research with the aim of improving natural language generation systems.

Acknowledgements

This research was supported by the European Research Council (Grant agreement No. 101039303 NG-NLG), the Charles University projects GAUK 40222 and SVV 260698, and the Apple NLU Research Grant for Heriot-Watt University and Charles University. Garanina’s work is supported by the Erasmus Mundus Master’s Programme “Language and Communication Technologies”.¹⁹ We acknowledge the utilization of resources provided by the LINDAT/CLARIAH-CZ Research Infrastructure (Czech Ministry of Education, Youth and Sports project No. LM2018101) and Peregrine HPC cluster (University of Groningen).

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. **Knowledge Graph Based Synthetic Corpus Generation for Knowledge-enhanced Language Model Pre-training**. 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 3554–3565.

Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Févry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged Saeed AlShaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir R. Radev, Mike Tian-Jian Jiang, and

Alexander M. Rush. 2022. **PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022 - System Demonstrations, Dublin, Ireland, May 22-27, 2022*, pages 93–104.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. **Constrained Decoding for Neural NLG From Compositional Representations in Task-oriented Dialogue**. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 831–844.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. **Table-to-Text: Describing Table Region With Natural Language**. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5020–5027.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. **Logical Natural Language Generation From Open-domain Tables**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7929–7942.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyu Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. **Logic2Text: High-fidelity Natural Language Generation From Logical Forms**. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 2096–2111.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. **HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1094–1110.

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2021. **EventNarrative: A Large-scale Event-centric Dataset for Knowledge Graph-to-text Generation**. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Longxu Dou, Guanghui Qin, Jinpeng Wang, Jin-Ge Yao, and Chin-Yew Lin. 2018. **Data2Text Studio: Automated Text Generation From Structured Data**. In *Proceedings of the 2018 Conference on Empirical*

¹⁹<https://lct-master.org>

- Methods in Natural Language Processing: System Demonstrations*, pages 13–18, Brussels, Belgium.
- Ondrej Dusek, David M. Howcroft, and Verena Rieser. 2019. [Semantic Noise Matters for Neural Natural Language Generation](#). In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 421–426.
- Christian Federmann. 2018. [Appraise Evaluation Framework for Machine Translation](#). In *COLING 2018, The 27th International Conference on Computational Linguistics: System Demonstrations, Santa Fe, New Mexico, August 20-26, 2018*, pages 86–88.
- Thiago Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 Bilingual, Bi-Directional Webnlg+ Shared Task Overview and Evaluation Results (Webnlg+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. [The WebNLG Challenge: Generating Text From RDF Data](#). In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133.
- Albert Gatt and Emiel Krahmer. 2018. [Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation](#). *J. Artif. Intell. Res.*, 61:65–170.
- Sebastian Gehrmann, Tosin P. Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh D. Dhole, Wanyu Du, Esin Durmus, Ondrej Dusek, Chris Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Rubungo Andre Niyongabo, Salomey Osei, Ankur P. Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. [The GEM Benchmark: Natural Language Generation, Its Evaluation and Metrics](#). *CoRR*, abs/2102.01672.
- Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2022. [Repairing the Cracked Foundation: A Survey of Obstacles in Evaluation Practices for Generated Text](#). *CoRR*, abs/2202.06935.
- Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. [TableGPT: Few-shot Table-to-text Generation With Table Structure Reconstruction and Content Matching](#). In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1978–1988.
- Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. [GenWiki: A Dataset of 1.3 Million Content-sharing Text and Graphs for Unsupervised Graph-to-text Generation](#). In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2398–2409.
- Mihir Kale and Abhinav Rastogi. 2020. [Text-to-text Pre-training for Data-to-text Tasks](#). In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020*, pages 97–102.
- Shankar Kantharaj, Rixie Tiffany Ko Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq R. Joty. 2022. [Chart-to-Text: A Large-scale Benchmark for Chart Summarization](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4005–4023.
- Johannes Kehrler and Helwig Hauser. 2013. [Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey](#). *IEEE Trans. Vis. Comput. Graph.*, 19(3):495–513.
- Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. [Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations](#). *Int. J. Comput. Vis.*, 123(1):32–73.
- Rémi Lebret, David Grangier, and Michael Auli. 2016. [Neural Text Generation From Structured Data With Application to the Biography Domain](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Sasko, Guntjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. 2021. [Datasets: A Community Library for Natural Language Processing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language*

- Processing: System Demonstrations, EMNLP 2021, Online and Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 175–184.
- Bill Yuchen Lin, Ming Shen, Yu Xing, Pei Zhou, and Xiang Ren. 2019. [CommonGen: A Constrained Text Generation Dataset Towards Generative Commonsense Reasoning](#). *CoRR*, abs/1911.03705.
- Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022. [PLOG: Table-to-logic Pre-training for Logical Table-to-text Generation](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5531–5546.
- Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. [Learning to Reason for Text Generation From Scientific Tables](#). *CoRR*, abs/2104.08296.
- Linyong Nan, Dragomir R. Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangu Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. [DART: Open-domain Structured Data Record to Text Generation](#). 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 432–447.
- Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. [Compare-Mt: A Tool for Holistic Comparison of Language Generation Systems](#). 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, Demonstrations*, pages 35–41.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. [Training Language Models to Follow Instructions With Human Feedback](#). In *NeurIPS*.
- Ankur P. Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. [ToTTo: A Controlled Table-To-text Generation Dataset](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1173–1186.
- 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 *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. [Multi-task Prompted Training Enables Zero-shot Task Generalization](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Elie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. 2022. [BLOOM: A 176b-parameter Open-access Multilingual Language Model](#). *CoRR*, abs/2211.05100.
- Mandar Sharma, Ajay Gogineni, and Naren Ramakrishnan. 2022. [Innovations in Neural Data-to-text Generation](#). *CoRR*, abs/2207.12571.
- Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. [Towards Table-to-text Generation With Numerical Reasoning](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 1451–1465.

- Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. [MVP: Multi-task Supervised Pre-training for Natural Language Generation](#). *CoRR*, abs/2206.12131.
- Craig Thomson, Ehud Reiter, and Somayajulu Sripada. 2020. [SportSett:basketball - A Robust and Maintainable Data-set for Natural Language Generation](#). In *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*, pages 32–40, Santiago de Compostela, Spain.
- Chris van der Lee, Chris Emmery, Sander Wubben, and Emiel Kraemer. 2020. [The CACAPO Dataset: A Multilingual, Multi-domain Dataset for Neural Pipeline and End-to-end Data-to-text Generation](#). In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020*, pages 68–79.
- Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Stephanie Schoch, Craig Thomson, and Luou Wen. 2023. [Barriers and Enabling Factors for Error Analysis in NLG Research](#). *North-eastern European Journal of Language Technology*, 9(1). Number: 1.
- Emiel van Miltenburg, Miruna-Adriana Clinciu, Ondrej Dusek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Emma Manning, Stephanie Schoch, Craig Thomson, and Luou Wen. 2021. [Underreporting of Errors in NLG Output, and What to Do About It](#). In *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021*, pages 140–153, Aberdeen, Scotland, UK.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krma Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. [Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5085–5109.
- 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.
- Yang Xiao, Jinlan Fu, Weizhe Yuan, Vijay Viswanathan, Zhoumianze Liu, Yixin Liu, Graham Neubig, and Pengfei Liu. 2022. [DataLab: A Platform for Data Analysis and Intervention](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 182–195, Dublin, Ireland.
- Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. [UnifiedSKG: Unifying and Multi-tasking Structured Knowledge Grounding With Text-to-text Language Models](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 602–631.
- Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2SQL: Generating Structured Queries From Natural Language Using Reinforcement Learning](#). *CoRR*, abs/1709.00103.

A ToTTo Header Highlighting

The original ToTTo dataset does not provide explicit highlighting for the row and column headers of the corresponding cells. The released table linearization script includes header linking, but the header extraction heuristic is prone to errors in some cases (see, e.g., tables 310 and 838 in the ToTTo development split). Therefore, we implemented a custom algorithm for incorporating header highlighting into our dataset version:

1. process raw ToTTo input: restore the rectangular representation of the table including the merged cells;
2. separate given headers into the column and row headers;
3. for each highlighted cell, highlight all column headers above it and all row headers to the left.

Out of 50 randomly selected tables with highlights different from the available ones, our version was correct and the original version incorrect in 36 cases. Out of the remaining 14 cases, the original version is correct while ours is not in 5 cases, and both versions are incorrect in 9 cases. However,

given the complex structure of some tables, our algorithm can sometimes fail to capture relevant row headers or mark several extra cells, which we plan to address in the future.

B Fine-tuned models

For the demo purposes, we have fine-tuned the following models using our example scripts:

- `t5-small` for Chart-To-Text, LogicNLG, ToTTo, WikiTableText;
- `t5-base` for DART, E2E, WebNLG;
- `t5-base` in a prefix-based multi-task setup on E2E and WebNLG, using custom linearization functions.

All models (individual and multi-task) were fine-tuned using `transformers` library. The parameters are the following:

- Epochs: 30 for individual models and 15 for multi-task,
- Patience: 5 epochs,
- Batch size: 16,
- Optimizer: AdamW,
- Learning rate: $1e-4$,
- Weight decay: 0,
- AdamW betas: 0.9, 0.999,
- Maximum input length: 512,
- Maximum output length: 512,
- Generation beam size: 3.

C User Interface

Figure 3 shows the interactive mode in the TABGENIE web interface. Figure 4 shows the spreadsheet for manual annotations generated using TABGENIE.

D System Requirements

Computational Requirements TABGENIE can run on a single-threaded CPU, although multiple threads can speed-up the initial dataset preprocessing.

Memory Requirements After pre-loading all the currently featured development sets into memory, TABGENIE consumes around 1 GB RAM.

Disk space Downloading all the datasets requires around 4 GB of disk space. The directory used for caching the datasets can be set using the `HF_DATASETS_CACHE` environment variable.

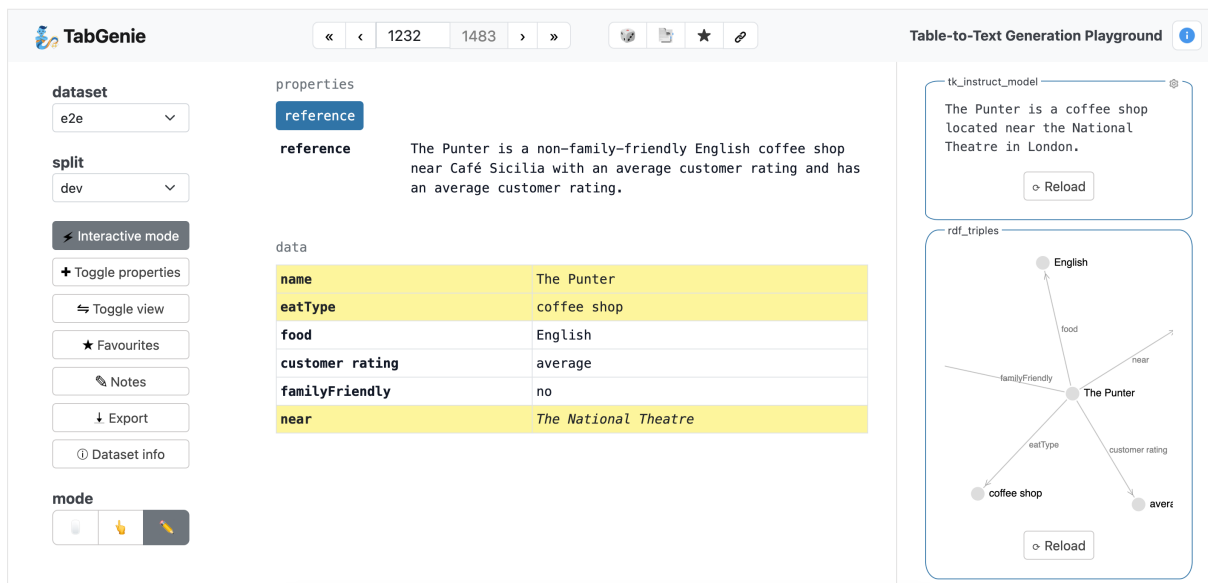


Figure 3: The interactive mode of the web interface in which the user (1) highlighted specific cells (the cells with the yellow background), (2) edited the input in one of the cells (“Café Sicilia” → “the National Theatre”), (3) re-generated the model output (see the top right panel). The figure also shows the graph visualization of the input key-value pairs.

	A	B	C	D	E	F	G	H
1	table_id	notes	property_name	property_value	table			
2	1309		reference	Adenan Satem was b	subject	predicate	object	
3			prediction	Adenan Satem was b	Abdul Taib Mahmud	successor	Adenan Satem	
4					Adenan Satem	birth place	Japanese occupation of British Borneo	
5					Abdul Taib Mahmud	residence	Sarawak	
6					Abdul Taib Mahmud	party	Barisan Ra'ayat Jati Sarawak	
7								
8	228		reference	Asam pedas is a food	subject	predicate	object	
9			prediction	Asam pedas is a food	Asam pedas	country	Malaysia	
10								
11	51		reference	Aleksandra Kovač's g	subject	predicate	object	
12			prediction	Aleksandra Kovac pe	Aleksandra Kovač	genre	Soul music	
13								
14	1518		reference	Chinabank, a publicl	subject	predicate	object	
15			prediction	Chinabank was foun	Chinabank	founding date	1920-08-16	
16					Chinabank	net income	15100000000	
17					Chinabank	number of locations	295	
18					Chinabank	foundation place	Manila	
19					Chinabank	type	Public company	
20								
21	563		reference	The main product of	subject	predicate	object	
22			prediction	Hypermarcas, locate	Hypermarcas	product	Drugs	
23					Hypermarcas	location	São Paulo	
24								
25	501		reference	The Asher and Mary	subject	predicate	object	
26			prediction	Asher and Mary Isab	Asher and Mary Isab	location	U.S. Route 83	
27					Asher and Mary Isab	National register of h	88002539	

Figure 4: The spreadsheet for manual annotations with a random sample of system outputs exported using TABGENIE.

An Efficient Conversational Smart Compose System

Yun Zhu, Xiayu Chen, Lei Shu, Bowen Tan, Xinying Song,
Lijuan Liu, Maria Wang, Jindong Chen, Ning Ruan

Google Inc.
yunzhu@google.com

Abstract

Online conversation is a ubiquitous way to share information and connect everyone but repetitive idiomatic text typing takes users a lot of time. This paper demonstrates a simple yet effective cloud based smart compose system to improve human-to-human conversation efficiency. Heuristics from different perspectives are designed to achieve the best trade-off between quality and latency. From the modeling side, the decoder-only model exploited the previous turns of conversational history in a computation lightweight manner. Besides, a novel phrase tokenizer is proposed to reduce latency without losing the composing quality further. Additionally, the caching mechanism is applied to the serving framework. The demo video of the system is available at <https://youtu.be/U1KXkaqr60g>. We open-sourced our phrase tokenizer in <https://github.com/tensorflow/text>.

1 Introduction

Online conversations are happening in every corner of the world in every second. People relies on different channels like daily chatting apps, e.g. Messages, WhatsApp, or online customer service to communicate with friends, families, colleagues and even acquaintance. Within conversational applications, efficient and smart assistant functions for users are long-desired. Smart compose (Chen et al., 2019) is a well-known smart writing feature that actively predicts the next couple of words in real-time to assist human writing. It saves user’s time by cutting back on repetitive idiomatic writing. In this paper, we build a smart compose system for chatting applications to improve user’s communication efficiency and enhance user experience. The outlook of the demo is shown in Fig 1. We assume two or multiple users are discussing some random topic in a chat room. And smart compose is working under the hood to suggest what to type next. The full demo is in <https://youtu.be/U1KXkaqr60g>.

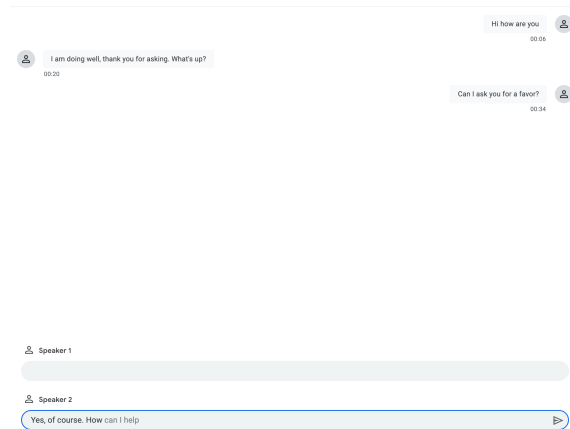


Figure 1: Demo of smart compose experiment. We type in “How” in the input box, the smart compose system suggests “can I help” in real time, which is based on the conversation context.

To the best of our knowledge, although smart compose is a well-known application, it has not been well-designed for conversation scenarios. One basic requirement of smart compose is the low inference latency. If the next words are not shown up in less than 100ms, users will likely type them themselves. And there are two challenges out of this. The first challenge is the conflict between latency and long conversation context. During the human-to-human conversation within the chatting applications, there are multiple turns of conversation from multiple users, making the conversation history informative. Attention over the whole conversation history would bring too much latency. However, simply ignoring the conversation history or using an average embedding of history to represent previous context (Chen et al., 2019) is not exploiting the rich information within the conversational flow between users. Therefore how to effectively and efficiently extract information from conversation history is the key consideration for model designing. The second challenge comes from the fact that users would prefer a longer sug-

gestion for conversation but generating a longer sequence is likely to involve more latency. In the auto-regressive generative model for example, the decoding time is linearly growing with the decoding steps. There is no clear solution on how to avoid the extra latency on the longer suggestion.

In this paper, we proposed a new solution to address above challenges and built an end-to-end cloud based smart compose system for human-to-human chatting applications. Our smart compose system can achieve high exact match rate with low latency (less than 100ms) under acceptable suggestion length. Our contributions are three-fold.

- We designed a novel architecture based on transformer-XL (Dai et al., 2019)¹ to effectively encode history and context information for composing. Also, it incorporates conversational features like user id and local position id to retrain the information effectively.
- We proposed a novel efficient tokenizer called phrase tokenizer which can increase the suggestion length for composing without sacrificing the quality and latency. This tokenizer can significantly reduce the latency of the system. We open-source our phrase tokenizer in <https://github.com/tensorflow/text> as an extension of the TensorFlow text library.
- We designed the cloud serving mechanism and addressed the practical issues such as caching.

2 Related work

One of the most successful applications for smart compose is Gmail (Chen et al., 2019), where the title of the email, the previous paragraphs, and the previous text of the current typing are combined to predict the next words. Following this success, similar models have also been used in other applications like document editing (e.g., Google Docs) or input method (e.g., Gboard). Although this feature is not new, previous work mainly focuses on general writing assist and not is specially designed for conversation scenarios.

Approach wise, there are two major categories. The first category is the dual encoder based model (Gillick et al., 2018; Karpukhin et al., 2020; Yang et al., 2019) which exploited contrastive learning:

¹In practice, we found transformer based system has higher quality compared to LSTM and thus we build our system with Transformer.

the predefined whitelist is built in advance, and the prefix tries to match the most possible one in the list. This kind of solution is also widely used in practical retrieval or recommendation systems. Although the dual encoder is extremely fast, the predefined whitelist limits the headroom for model quality from smart compose.

The second category is language modeling, a fundamental and indispensable component of many natural language processing. Recently it is becoming even more powerful with a larger model size (Floridi and Chiriatti, 2020; Thoppilan et al., 2022; Chowdhery et al., 2022). However, with the latency requirement, models should be designed smaller and more light-weighted for the smart compose (Van et al., 2020; Ciurumelea et al., 2020; Chen et al., 2019). Compared to dual-encoder, it removes the dependency on the whitelist and could achieve better matching accuracy. However, as it is involved with the autoregressive decoding process, the latency is generally higher and positive relative to the number of decoding steps. Although non-autoregressive based approaches (Zou et al., 2021) could potentially reduce the latency for text generation, it has been well-known to suffer from quality regression.

Our smart compose system is more conversational oriented and can achieve the best trade-off between quality and latency. Our model share the benefit of higher quality with the language modeling, but keep the latency low with proposed techniques.

3 Model Design

In this section, we will outline the modeling design. Firstly, we will first introduce the conversational inputs for the model and what features we are extracting from the conversation history. Then we will go through the model architecture used to take the conversational input and output the predicted suggestions. Finally, we will discuss how the loss is calculated during training and some implementation details.

3.1 Conversational Inputs

Our smart compose is based on conversational data, and our data processing strategy is to process the conversation with the sliding window of fixed size N , which is the number of conversation turns (in practice, we choose N equals 10). We tokenize all N sentences and concatenate the N turns together

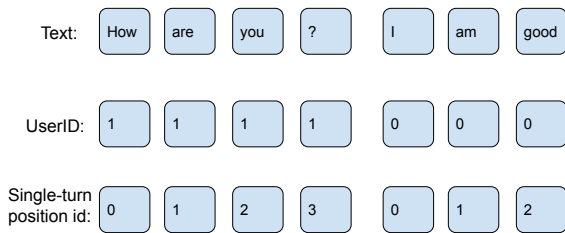


Figure 2: Example of conversational features.

to make a single input. Since the conversation could happen between multiple people, we also introduce the user id into the model input. The user id is associated with every token inside each turn conversation. Besides, we make use of the single turn position id. Different from the normal position embedding (Vaswani et al., 2017), which is calculated for all the input sequences, the single-turn position id is only for a sub-section of the input sequence. Furthermore, the N single-turn position id is concatenated together. The conversational features are in Fig 2. We found that using the user id and single-turn position id could achieve lower perplexity.

3.2 Model Architecture

We use separate embedding matrices to process the text id, user id, and single-turn position id. Specifically, we have separate embedding matrices for text-token, user-id, and single-turn position id. We add embedded vectors together as the final input embedding.

We feed this input embedding to our transformer decoder. Our architecture is a decoder-only Transformer XL (Dai et al., 2019) model. Note that the relative position is still in place, although we have an extra single-turn position id. The overall architecture is shown in Figure 3.

To reduce the latency, we adopted local attention (Beltagy et al., 2020) instead of full attention. We found that even with limited look-back length, the prediction can be accurate enough. By joint using Transformer-XL and local attention, the transformer architecture is able to perform with lower computation, and similar to LSTM it can pass the state during processing the sequence. This helps capture the context information with little overhead. More details will be discussed in Section 5.

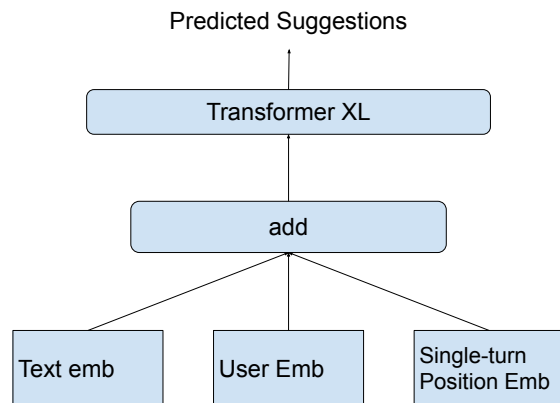


Figure 3: Model Architecture used for conversational smart compose.

3.3 Loss and Training Framework

We right-shift the input as the target and use the simple cross-entropy loss for the language model training. To be aligned with our sliding window data processing, we use a loss mask that only masks on the "last" turn. This is to ensure we are re-calculating the cross entropy for each turn. Our training framework is based on Tensorflow Model Gardens (Banna et al., 2021).

4 Phrase Tokenizer

Autoregressive decoding is the primary way of generating text with a Transformer decoder. One drawback of this mechanism is the latency issue: with N output units, the model will forward pass N times in a sequential manner. In this paper, we design a token that is beyond a single word (a phrase with n -gram) and a tokenizer that emits a group of units at a time, with the fallback mechanism to emit a single unit so that we can decrease the N (i.e., number of output tokens) here to reduce the latency. Note that the basic element of the phrase is a word following the findings of (Chen et al., 2019) that the word-based vocabulary can achieve better performance for smart compose. Nevertheless, our approach can also be applied to word-piece or sentence-piece models.

4.1 Ngram phrase Tokens

We build the Ngram dictionary based on the text. By definition, we treat P_1 as the set of unigrams with higher frequency. Likewise, to build P_2 we collect all word bigrams that occur in the training set with higher frequency (e.g., greater than a certain threshold) and augment this set of bigrams

with the P_1 . Moreover, we can build P_3 and P_k similarly.

4.2 Random Tokenizing Process

Once we have the Ngram phrase token set ready, the intuitive solution for tokenization is to tokenize the input text string from left to right to maximize the current N in NGram. This greedy method, however, can not work well in the smart compose application. The reason is that a user can stop at any word. Using the example of the phrase “How are you”, if the model always treats this as a single token during training, it can never make the right next word prediction when the user types “How are”. Therefore we should consider both phrase and individual word simultaneously.

To solve this problem we designed a random picking process when tokenizing the input text. The main idea is whenever there is multiple way to tokenize a phrase or sentence, we randomly pick one of them. We still enforce the left-to-right manner here. To efficiently find all the possible phrase from certain point, we leverage the Double Array Trie (Yata et al., 2007), which provides the function of iteratively finding the prefix match of a given text string. We introduce the additional parameter prob as input, which denotes the probability of choosing the current prefix match as opposed to finding a longer one. This randomness ensures that the model can see both phrase tokens and single-word tokens. We summarize our tokenization process in Algorithm 1.

5 Serving System Design

We will discuss the serving system design and heuristics in this section. Specifically, we will introduce how we design the inference graph to take advantage of the caching of conversation history. Besides, we will illustrate the server-side components of the system.

5.1 Inference Heuristics

To alleviate the burden of the server, we move complicated logic into the inference graph and serve this inside a standard servo. Specifically We put both the phrase tokenizer and the greedy (beam) search process inside the inference graph. We also customized the greedy or beam search process. To make sure <UNK> will not become the output, we regard the <UNK> token as a <EOS> token and update the log probability as the previous one when

Algorithm 1 Random-picking tokenization

```

Require: raw_input
Require: A Double Array Trie that contains all phrases trie
Require: The prob of emitting the phrase prob
0: tokens = []
0: wordlist = WhiteSpaceTokenizer(raw_input)
0: concat wordlist with space as input
0: len = len(input)
0: while  $do$   $i < len$ 
0:   matches = trie.IteratePrefixMatches(input(i:))
0:   for  $do$   $match \in matches$ 
0:     if RandomGen() > prob then
0:        $i+ = match.length$ 
0:       tokens.append(match.id)
0:     Break
0:   end if
0:   end for
0: end while
0: Output tokens =0

```

stopping at the <UNK>.

Since the conversation for chat is usually multi-turn and short, we use Transformer XL to encode the previous turns into model states and provide two sub-inference graphs to handle the caching and suggestions. The first sub graph takes in the previous conversation history and encode the states as

$$\text{States} = \text{Model}(\text{ConversationHistory}) \quad (1)$$

The States is tensors of the intermediate output for TransformerXL. We will cache this States and do compose for any current user input as.

$$\text{Suggestions} = \text{Model}(\text{CurrentInput}, \text{States}) \quad (2)$$

5.2 Serving Flow

The serving infra of smart compose consists of the API front-end, prediction, and cache layers. The front-end layer receives conversational data, truncated according to sliding window size N, separated into the conversation history storing the previous conversation turns, and the current input storing the current message being typed. Such data can then be packed into sequential servo prediction requests sent to the prediction layer for states and suggestions. Furthermore, by adding the cache layer, we can further store the value of states keyed by their conversation history so that when multiple

smart compose suggestions are requested at various keystrokes of the same current input message, the prediction request for a state can be swapped by a cache retrieval as the conversation history stays the same, and hereby reduce the latency.

6 Experiments

6.1 Data and Model Setup

Our experimental data is based on conversational data collected in existing conversational apps. We use the previous 10 conversation as conversation history and restrict the max sequence length as 256 tokens. For the model configuration, we used the the transformer with 8 attention heads. The input dimension as well as model dimension is set to 256 and the dimension of feed-forward network is set to 1024. We did not find dropout useful, so the models are trained without dropout. All the models for experiments presented in this paper are trained on 8x8 Dragonfish TPU with a per-core batch size of 32 (effective batch size of 4096). The learning rate schedule is ramped up linearly from 0 to $8.0e-4$ during the first 10,000 steps, and then it decays exponentially to zero till 500,000 steps. We use CPU for serving. The total model size is around 5MB.

6.2 Results

We mainly use the exact match to measure the accuracy of the model. For latency, the cpu based servo load-test is used.

6.2.1 XL and Local Attention

We first evaluate how much XL and local attention could help in the smart compose tasks. In the composing task, the latency is related to the number of decoding steps. Meanwhile, for the quality side, the more decoding steps, the worse the exact match.

Using local attention can effectively reduce latency because of two reasons. First, it will have a smaller payload for RPC requests, as the local attention span decides the tensor size of model states. Secondly, it reduces the model computation, especially the computation of self-attention. Table 1 compares the latency with different attention spans. We found that local attention can effectively reduce the latency by a large ratio. However using the local attention does not give much penalty for exact match. As shown in Table 2.

	Server latency (ms)	Total latency (ms)
full attn	19.3	23.4
attn=32	14.1	16.6
attn=8	12.6	13.4

Table 1: Comparison of Latency of transformer XL with location attention to its counterparts. Attn means the "length of local attention".

	DS=1	DS=2	DS=3	DS=4	DS=5
full attn	88.6	69.1	56.5	49.4	45.9
attn=32	88.5	69.0	56.5	49.4	45.8
attn=8	87.4	68.0	54.6	48.4	44.7

Table 2: Comparison of Exact Match of transformer XL with location attention to its counterparts. "DS" means "decoding steps".

6.2.2 Phrase Tokenizer

We evaluate the effectiveness of the proposed phrase tokenizer in this section. To better illustrate the suggestion length, another critical factor for smart compose, we will look at two more additional metrics besides exact match. The first one is average length, which measures the length of the phrase (i.e., number of words) in the prediction when it is a correct prediction. The second one is effective length. This is also considering the correctness when calculating the prediction length, i.e., if the prediction is wrong, we will treat it as 0. Please note that these two metrics only give partial evidence of the composing quality.

For both the word level and phrase level tokens, we use the vocabulary of 60,000 tokens. For phrase level, we combine single words and phrases: we pick 30,000 single-word tokens and 30,000 phrase tokens which have 2-5 single words. We found that combining them can achieve the best quality.

We compare two tokenizers using the same model architecture, the word-based tokenizer, and the phrase-based tokenizer. The results are summarized in Table 4. For the word-based compose, we decode up to 5 steps. While for phrase-based compose, we only need to decode up to 2 steps. When the phrase-based model decodes two steps, and the word-based model decodes 4 or 5 steps, they achieve similar prediction length and exact match performance. In other words, with a phrase tokenizer, decoding two steps has the same quality effect as decoding 4-5 steps with a word tokenizer.

Prefix	Phrase tokenization	Steps	Word tokenization	Steps
Hi	"How are you"	1	"How", "are", "you"	3
I would	"like to thank you"	1	"like", "to", "thank", "you"	4
May I	"know when you have", "time"	2	"know", "when", "you", "have", "time"	5
Don't	"worry about", "it"	2	"worry", "about", "it"	3
How	"can I help you"	1	"can", "I", "help", "you"	4
We	"have to", "send it", "to", "him"	4	"have", "to", "send", "it", "to", "him"	6
I'll be	"more than happy to", "help you"	2	"more", "than", "happy", "to", "help", "you"	6
Can you	"tell me", "what is", "wrong"	3	"tell", "me", "what", "is", "wrong"	5

Table 3: Case Study of Phrase tokenizer VS Word tokenizer. When using the phrase tokenizer, fewer decoding steps is required to compose the same length as the word level tokenizer.

Metric	Tokenization	DS=1	DS=2	DS=3	DS=4	DS=5
Exact Match	Word	87.4	68.0	54.6	48.4	44.7
	Phrase	67.2	46.2	-	-	-
Average Length	Word	1	1.991	2.879	3.692	4.43
	Phrase	2.132	4.384	-	-	-
Effective Length	Word	0.874	1.345	1.572	1.786	1.98
	Phrase	1.432	2.027	-	-	-

Table 4: Performance comparison between phrase tokenizer and word tokenizer. The phrase tokenizer can achieve a similar composing length and exact matching rate with fewer decoding steps. "DS" refers to "decoding steps".

Latency (ms) / Steps				
1	2	3	4	5
3.9	7.5	9.5	12.6	14.8

Table 5: Latency Per steps.

However, decoding only two steps can achieve significantly lower latency, as shown in Table 5.

To illustrate how the phrase tokenizer behaves differently from the word level tokenizer, we show examples in Table 4. Our phrase tokenizer contains common phrases for daily conversations, such as "How are you", "can I help you", and "more than happy to". With these phrases as tokens, our solution can largely reduce the decoding steps and the latency. Furthermore, besides common phrases, phrase tokenizer contains single words as tokens, for example, "to", "it", "him", "wrong". These single-word tokens extend the granularity and diversity of tokens and ensure composing quality.

6.3 Demonstration

In this section, we demonstrate how the conversational smart compose system works. More details are described in the demo video. The user input

interface is shown in Figure 1. While a user can type messages in the text box, the smart compose model will provide real-time suggestions. If the user is satisfied with the suggested text, the user can accept this suggestion; if not, the user can continue typing, and new suggestions will be shown along with user types. The lower latency comes from the model architecture, phrase tokenizer, and caching mechanism during serving.

7 Conclusion

In this paper, we demonstrate a simple and efficient conversational smart compose system. Our model is adopted from Transformer XL and effectively encodes history and context information to be used during composing. We proposed a novel and efficient tokenizer called phrase tokenizer to reduce latency for composing applications. Efficient serving heuristics and caching is also applied. With all the merit above, our demo achieved long, accurate and real-time typing suggestions for the conversation. We open-sourced the phrase tokenizer as a part of tensorflow text library.

References

- Vishnu Banna, Akhil Chinnakotla, Zhengxin Yan, Anirudh Vegesana, Naveen Vivek, Kruthi Krishnappa, Wenxin Jiang, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. 2021. An experience report on machine learning reproducibility: Guidance for practitioners and tensorflow model garden contributors. *arXiv preprint arXiv:2107.00821*.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295.
- 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*.
- Adelina Ciurumelea, Sebastian Proksch, and Harald C Gall. 2020. Suggesting comment completions for python using neural language models. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 456–467. IEEE.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.
- Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.
- Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.
- 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*.
- 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 e-prints*, pages arXiv–2201.
- Hoang Van, David Kauchak, and Gondy Leroy. 2020. Automets: the autocomplete for medical text simplification. *arXiv preprint arXiv:2010.10573*.
- 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.
- Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. 2019. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.
- Susumu Yata, Masaki Oono, Kazuhiro Morita, Masao Fuketa, Toru Sumitomo, and Jun-ichi Aoe. 2007. A compact static double-array keeping character codes. *Information processing & management*, 43(1):237–247.
- Yicheng Zou, Zhihua Liu, Xingwu Hu, and Qi Zhang. 2021. Thinking clearly, talking fast: Concept-guided non-autoregressive generation for open-domain dialogue systems. *arXiv preprint arXiv:2109.04084*.

Which Spurious Correlations Impact Reasoning in NLI Models? A Visual Interactive Diagnosis through Data-Constrained Counterfactuals

Robin Chan¹ Afra Amini^{1,2} Mennatallah El-Assady^{1,2}
¹ETH Zürich ²ETH AI Center
chanr@ethz.ch {afra.amini, melassady}@inf.ethz.ch

Abstract

We present a human-in-the-loop dashboard tailored to diagnosing potential spurious features that NLI models rely on for predictions. The dashboard enables users to generate diverse and challenging examples by drawing inspiration from GPT-3 suggestions. Additionally, users can receive feedback from a trained NLI model on how challenging the newly created example is and make refinements based on the feedback. Through our investigation, we discover several categories of spurious correlations that impact the reasoning of NLI models, which we group into three categories: Semantic Relevance, Logical Fallacies, and Bias. Based on our findings, we identify and describe various research opportunities, including diversifying training data and assessing NLI models' robustness by creating adversarial test suites.



<https://dcc.lingvis.io>

1 Introduction

The availability of crowdsourced large-scale datasets has been influential in the field of natural language processing. These datasets have empowered advancements in a wide range of downstream tasks, including the natural language inference (NLI) task (SNLI; Bowman et al., 2015). While being influential, crowdsourcing frameworks can introduce artifacts, biases, and spurious correlations that can negatively impact the robustness and out-of-domain generalization of the models that are trained on such datasets (Jia and Liang, 2017; McCoy et al., 2019).

A **spurious correlation** exists when a feature correlates with the target label while there is no causal relationship between the feature and the label. For example, the fact that a sentence includes the word “amazing” (as a feature) might correlate with a positive sentiment but does not *cause* the sentiment label to be positive, as one can imagine

crafting a sentence like “the amazing concert was ruined by the terrible acoustics in the venue”, which has a negative sentiment. It has been shown that such spurious correlations exist in crowdsourced datasets (Gardner et al., 2021), and this will prevent models that are trained on these datasets from performing on adversarial or out-of-domain test sets (McCoy et al., 2019).

One approach to prevent a model from relying on spurious correlations between a feature and the label is to break such correlations by providing **counterfactuals** during training. In this context, counterfactuals are data points that contain the feature but have a different label. Following our previous example, “the amazing concert was ruined by the terrible acoustics in the venue” is a counterfactual sentence since it contains the word “amazing” but has a negative sentiment. Augmenting datasets with counterfactuals can break the spurious correlations and help the model to generalize to out-of-domain examples. However, generating counterfactuals is challenging; it involves first identifying noncausal features that correlate with the label, i.e., spurious correlations, and then generating the counterfactuals for a given feature.

One simple approach to generate counterfactuals is through minimal edits. In this approach, the first step—identifying spurious correlations—is bypassed. Therefore, counterfactuals are generated without targeting any specific feature. To generate a counterfactual, existing data points in the dataset are edited minimally such that they have a different label compared to their original one. While such an approach is scalable and can be effective in certain scenarios (Khashabi et al., 2020), creating counterfactuals through minimal edits does *not* necessarily improve the generalization of models and might even hurt the performance (Huang et al., 2020). Therefore, there is a need for a more nuanced and innovative approach to counterfactual generation.

In this paper, we propose a data-centric approach

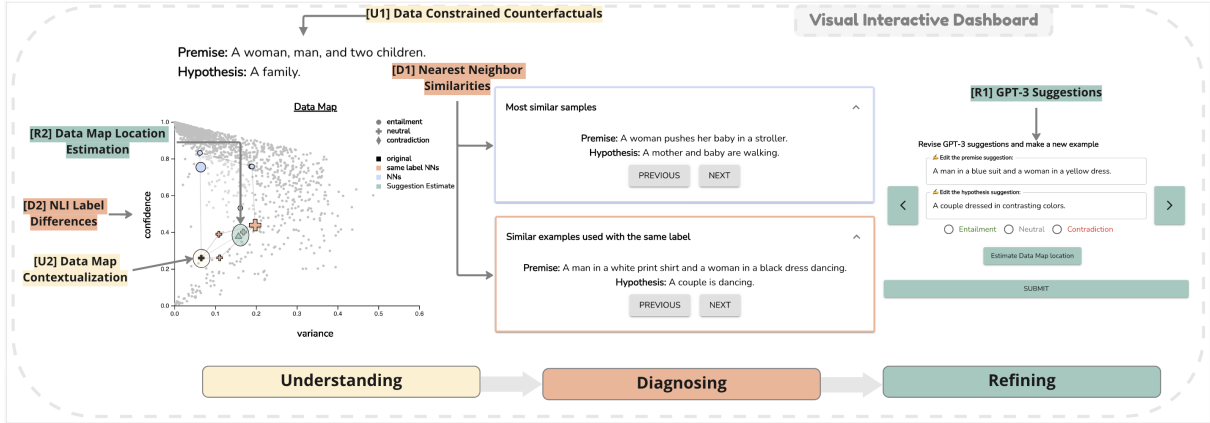


Figure 1: The three main phases in our interactive dashboard. In the first step, [U1], [U2], the user understands the main data point and the prediction of the model on that data point. In the second step [D1], [D2], the user diagnoses the similarities and differences between other data points in the dataset and the main data point. In the last step, [R1], [R2], the user revises GPT-3 suggestions using the feedback from the model and submits a counterfactual.

to counterfactual generation. First, we identify existing counterfactuals in datasets, which we term **data-constrained counterfactuals** (DCC). Second, using our interactive dashboard, we diagnose features that spuriously correlate with their label by comparing and contrasting the DCC with other data points in the dataset. Lastly, we generate a diverse set of counterfactual examples with the help of GPT-3 (davinci-003; Brown et al., 2020).

Overall, our dashboard offers a human-in-the-loop, or more generally, a mixed-initiative approach. A user can diagnose spurious correlations and common patterns that result in NLI models’ inability to predict the labels correctly. Finding such weak spots can provide ways to improve the NLI model. Furthermore, after the user has generated a set of new counterfactuals, the NLI model can give feedback on how valuable each counterfactual is by expressing its uncertainty in predicting the sample’s annotated label. This can help the user to revise the counterfactual and improve the usefulness of the generated set.

While our dashboard can be extended to various tasks in natural language processing, we focus on the NLI task in this work. Using our dashboard, we find a variety of features that correlate spuriously with labels. We categorize these features into three categories, which we name: Semantic Relevance, Logical Fallacies, and Biases. We further find a category of samples that are annotation artifacts. Based on these findings, and with the help of our dashboard, one can create novel counterfactuals to assess the robustness of NLI models or use them to augment training sets.

2 Preliminaries

Before introducing our approach, we first go through some preliminaries. We briefly describe the NLI task and a tool called Data Maps.

Natural Language Inference (NLI). We employ our dashboard for the NLI task. The task is to determine whether a **premise** *entails*, *contradicts*, or is *neutral* to a **hypothesis** (Condoravdi et al., 2003; Dagan et al., 2006; Bowman et al., 2015). As with many other NLP tasks, neural NLI models have been shown to rely on spurious correlations (Gardner et al., 2021). For example, they often predict *contradiction* when the hypothesis contains the word “not”. To obtain some hints on whether a model is relying on spurious correlations, we use data maps, which we describe next.

Data Maps. Swayamdipta et al. (2020) propose a tool called **Data Maps** to diagnose the characteristics of datasets with respect to a model’s behavior during training. They propose two axes to locate each training data point in two dimensions. First, the **confidence** is defined as the average probability that the model assigns to the *true* label throughout training checkpoints, and second, the **variability** of its confidence across the checkpoints. They identify three different regions in data maps: i) a region consisting of data points where the model has high confidence with low variability, i.e., **easy to learn** data points, ii) a region consisting of data points where the model’s confidence on the true label fluctuates a lot (high variability), i.e., **ambiguous** data points, and iii) a region where the model has low

confidence on the true label with low variability, i.e., **hard to learn** data points.

In this paper, we employ data maps at two stages. First, in §3 we discuss how to use data maps to locate DCCs. Second, we incorporate data maps in our interactive dashboard and further provide estimates of the location of newly created data points in the data map. Such an estimate gives early feedback to the user on how challenging it could be for the model to predict the label of the generated counterfactual. The user can then act on this feedback by revising the counterfactual.

3 Data-Constrained Counterfactuals

In this work, we propose to start with finding existing counterfactuals in datasets. We will later use these counterfactuals in our dashboard §4 to find spurious features and generate new data points.

A data-constrained counterfactual is a data point that shares some features with other data points in the dataset but has a different label. Further, we want to make sure that the model is sensitive to the spurious correlation. Therefore, it *should not* be easy for the model to label a DCC correctly. We provide the following formal definition of data-constrained counterfactuals.

Definition 1. *A data point is a data-constrained counterfactual (DCC) when it satisfies two conditions: i) there exists other data points in the training set that are similar to this data point but have a different label, and ii) it is not easy for the model to label it correctly, i.e., it falls into either the hard-to-learn or the ambiguous region in the data map.*

This definition relies on a notion of similarity; thus, to identify DCCs we need to provide a similarity metric between data points. Following Liu et al. (2022), we define the similarity between data points as the cosine similarity between the [CLS] embedding of data points given by the underlying model. This will give us a tractable measure to find similar data points in large datasets without any manual inspection of the data.

A caveat to Def. 1 is that many data points in the hard-to-learn region have been found to be mislabeled (Swayamdipta et al., 2020). To filter out samples that are likely to be mislabeled, we only select samples that have multiple annotations, where a large¹ majority of annotators agree on the label.

¹ $\geq 75\%$, as most multiple-annotated SNLI samples have four label annotations.

4 Visual Interactive Dashboard

In this section, we describe the tasks that users can perform during the interactive counterfactual generation process. We categorize these tasks using the *explAIner* framework (Spinner et al., 2020).

4.1 Understanding

First, the user is provided with enough information and supporting visuals to *understand* the DCC that is being selected. This involves two tasks, explained below.

[U1] Data-Constrained Counterfactuals. The premise, hypothesis, and label of the DCC are shown to the user. This ensures that the user can get an initial understanding of the example and the annotators’ reasoning.

[U2] Data Map Contextualization. The ground truth labels of the selected DCCs are inherently hard for the model to predict (see Def. 1). Therefore, it is helpful for the user to understand how the model reasons about the data point, i.e., how likely it is that the model predicts the correct label (confidence) and how often its prediction varies across different checkpoints (variance). To this end, we locate the selected data point in the data map (see Fig. 1 black data point) and visualize the data map in our dashboard.

4.2 Diagnosing

Next, we aim to diagnose the reason that the DCC ends up being a counterfactual. As mentioned earlier, we aim to find features that correlate spuriously with the label. To find common features between the DCC and other data points in the sentence, we visualize similarities and differences between the DCC and other data points in the dataset. This involves performing two tasks explained below.

[D1] Nearest Neighbor Similarities. We show two different sets of sentences in separate boxes and locate both sets in the data map. First, the set of sentences that are most similar to the DCC (in the blue box in Fig. 1). By definition (Def. 1), the most similar data points will have a *different* label compared to the selected data point. By comparing the DCC with the most similar data points, one might be able to find structures or patterns that are shared between the two. Those can be features that spuriously correlate with the label. Second, we depict the set of most similar data points with *the same* label as the data point (orange box in Fig. 1).

There might be more than one DCC breaking the spurious correlation in the dataset, and visualizing similar data points with *the same* label can help the user discover such examples and their similarities to the DCC. In sum, investigating the similarities and differences between these two sets will help the user to diagnose potential spurious features that are shared between the sets and correlate with the label.

[D2] NLI Label Differences. We are interested to determine which sentences in the training dataset may have influenced the DCC being mislabeled. For very similar samples, the labels of the nearest [CLS] neighbors are a strong indication of what the model would predict for the seed sample. Therefore, we visualize the label of nearest neighbors in the data map using three distinct shapes.

4.3 Refining

We will assist the user to create counterfactuals similar to the DCC, by pulling suggestions from GPT-3. The user can then refine the suggestion based on the feedback from the model.

[R1] GPT-3 Suggestions. Following (Liu et al., 2022), we use similar sentences with *the same* label to prompt GPT-3 and create suggestions. Ideally, GPT-3 would find the reasoning pattern and generate a valid counterfactual sentence. However, as one can imagine GPT-3 might fail to generate a valuable sample for several different reasons, e.g., it might generate an example that is semantically close to the DCC but the reasoning is not aligned with the DCC. Another reason would be to generate an example that is easy for the model to learn. Therefore, we ask the user to refine this new example before adding it to the dataset.

[R2] Data Map Location Estimation. One of the common errors with GPT-3 suggestions is that the suggestion might be easy for the model to learn. To filter those suggestions, after labeling the example, the user can request an estimate of the data map location. To ensure low latency for estimating the data map location of new examples, we do *not* re-train the model. Instead, we receive the label from the user and use the saved checkpoints to measure the confidence of the model on the true label and its variance across the checkpoints. The user can then iteratively refine the example if it ends up in the easy-to-learn region.

Instruction	Write a pair of sentences that have the same relationship as the previous examples.
Few-shot examples	Examples: 1. A man in a white print shirt and a woman in a black dress dancing. <i>Possibility</i> : A couple dancing. ... 5. A woman, a man and a child. <i>Possibility</i> : A family.
Empty, to be filled by GPT-3	6.

Figure 2: Example of GPT-3 few-shot prompting. The few-shot examples are the nearest neighbors with the same label as the DCC, ordered in increasing DCC similarity, and finally, the DCC. The word setting the premise in context with the hypothesis can be either *Implication*, *Possibility*, or *Contradiction*, depending on whether the DCC is labeled *entailment*, *neutral*, or *contradiction*.

5 Experimental Setup

The components of the dashboard are shown in Fig. 1. The filtering of potential DCCs described in section §3 was performed on the SNLI dataset (Bowman et al., 2015).² We compute the nearest neighbors of the DCCs according to the cosine similarity between the [CLS] embeddings extracted from a ROBERTA-large model (Liu et al., 2019) trained on SNLI. The data map was generated following (Swayamdipta et al., 2020), where six end-of-epoch checkpoints of the SNLI ROBERTA-large model were used to estimate the data map location.

Suggestions are generated by few-shot prompting the GPT-3 davinci-003 model (Brown et al., 2020) using four nearest neighbors to the DCC with the same label, exactly following (Liu et al., 2022), as they argue that the model may employ similar reasoning for such nearest neighbors. An example of such a prompt is shown and described in Fig. 2.

6 Findings

By interacting with the dashboard ourselves in multiple sessions, we find interesting patterns and many DCC instances following those patterns. In this section, we provide a categorization of our findings.

²DCC definition relies on having a large set of samples with multiple annotations, which is available in SNLI dataset.

We find three high-level features that correlate spuriously with the label, which we name: Semantic Relevance, Logical Fallacies, and Bias. Furthermore, we discovered another category that surfaces artifacts in the data collection procedure. Next, we will go through and explain each category, and further, provide some examples.

6.1 Semantic Relevance

We find many instances in the dataset where the hypothesis is the rephrased version of the premise. Clearly, in those cases, the gold label is entailment, e.g., (1).

- (1) *Premise:* A man in blue shorts and a t-shirt is slicing tomatoes on a dining table. **entails**
Hypothesis: A man prepares tomatoes by slicing them at the table.

However, if such examples dominate the dataset, a trained model might associate the entailment label to *any* premise and hypothesis that are semantically relevant to each other. The semantic relevance is a spurious feature, as one can imagine counterfactual examples where the premise and hypothesis are semantically related but the premise *does not* entail the hypothesis. One DCC that contains this feature is (2).

- (2) *Premise:* A large group of people are walking towards something, and most of them have backpacks. **is neutral to**
Hypothesis: A group of people move toward something that *requires* the use of a backpack.

In this example, while premise and hypothesis are semantically related, the word “requires” in the hypothesis makes the hypothesis to *neutral* to the premise, while the NLI model predicts the *entailment* label.

6.2 Logical Fallacies

Another common pattern we find in the dataset is hypotheses that become *neutral* to the premise by mentioning extra details.

- (3) *Premise:* A woman in a black dress and flat shoes holds her head as she waits to cross the street. **is neutral to**
Hypothesis: The woman is carrying a purse.

For example, in (3) the premise describes the appearance of a woman but does not mention any-

thing about whether she is carrying a purse. Therefore, the hypothesis is referring to an extra piece of information that was not mentioned in the premise and thus, is neutral to it. If such examples dominate the dataset, a trained model might associate any extra information in the hypothesis with a neutral label. However, in some scenarios, the presence of logical clues in the premise will result in a different label. Such a DCC in the data is shown in (4).

- (4) *Premise:* A man wearing *only* red pants does a trick on a ladder. **contradicts**
Hypothesis: The man is wearing a black shirt.

In this example, while the premise does not directly talk about whether the man is wearing a black shirt or not, the word “only” indicates that the hypothesis is in fact, false. However, the NLI model predicts the *neutral* label.

6.3 Biases

As with many other datasets, NLI datasets contain instances of different sorts of biases. Gender stereotypes in professions are one example.

- (5) *Premise:* A wrestler is jumping off of the ring to hit his competitor. **is neutral to**
Hypothesis: Two men are competing in a wrestling match.

In the above example (5), while there is no mention of the gender of wrestlers in the premise, the model predicts that the hypothesis *entails* the premise. This could be due to the fact that wrestling is stereotypically associated with men.

- (6) *Premise:* A woman, man, and two children. **is neutral to**
Hypothesis: A family.

Another example is (6), where we do not know the woman, man, and two children that the premise is describing are in fact a family. However, the model predicts *entailment* as the label for this example.

6.4 Artifacts

The last category of examples is the existing artifacts in the dataset that surfaces in our dashboard. We find several examples where the hypothesis is completely irrelevant to the hypothesis, but their labels are inconsistent and often wrong.

- (7) *Premise:* A child and woman exchange glances. **contradicts**
Hypothesis: a bird was on rocks.
- (8) *Premise:* A little child playing in water with a hose. **entails**
Hypothesis: a bird was on rocks.

While both examples (7) and (8) should have a *neutral* label, they are labeled as contradiction and entailment.

7 Discussion

The visual interactive dashboard for diagnosing spurious correlations and counterfactual generation can open up research opportunities in the following domains:

Bi-Directional Explanation of Reasoning Patterns. Our dashboard opens up a possibility for efficient collaboration between humans and AI. AI can help humans to find and group similar structures. As can be seen in our dashboard, similarities in the representation space of NLI models often capture similar structures. On the other hand, humans can explain the reasoning to AI. This can happen by generating new examples that follow a particular line of reasoning that is challenging for the AI model to learn, which can result in improving AI models.

Diversifying Training Data based on DCC. Receiving an estimate of model confidence during refinement ([R2]) enables the user to understand and pinpoint the patterns that pose a challenge to the model. Given the user has established such an understanding, they can produce samples that target a specific reasoning pattern. Further, GPT-3 suggestions assist the user by providing a diverse set of examples that follow the desired reasoning pattern. Therefore, the process allows us to augment potentially biased training datasets with a large, diverse set of counterfactuals. Conducting a thorough investigation, including large-scale expert annotation, model-retraining, and benchmarking is still required and will be part of future work.

Towards more Robust NLI Models. The counterfactual samples generated using our dashboard can be used as adversarial test suites for evaluating existing models. As a proof-of-concept, we generate a small set of such samples through our dashboard to evaluate a model trained on WaNLI

data (Liu et al., 2022),³ which itself was trained to be more robust and results in state-of-the-art results on various NLI test suites. The WaNLI model only achieves an accuracy of around 30% on our generations. This hints at the potential of our proposed annotation workflow for generating test suites to evaluate the robustness of NLI models.

8 Related Work

Other tools have been proposed for counterfactual generation. For example, POLYJUICE (Wu et al., 2021) introduces an automated counterfactual generation based on minimal edits. Counterfactuals are created using a fixed set of control codes to edit the existing sentences in the dataset.

Further, systems have been developed for mixed-initiative adversarial sample generation. ANLI (Nie et al., 2020) introduces an adversarial sample generation framework, where annotators are tasked to write hypotheses that may fool the model for a given context (i.e., a premise and label). Following a similar framework, Dynabench (Kielbaso et al., 2021) presents a more general-purpose dashboard for adversarial generation using model predictions and explanations as feedback to the user.

Compared to the methods described above, our proposed approach aims to first diagnose potential spurious correlations through DCCs, and then generate counterfactuals based on the found spurious correlations via prompting large language models. Furthermore, our dashboard gives feedback to users during the refinement stage by providing them with data map estimates for newly generated counterfactuals.

9 Conclusion

We present a dashboard to diagnose spurious correlations and artifacts that an NLI model may have acquired during training. We first provide a systematic approach to find data-constrained counterfactuals, i.e., existing counterfactuals in the dataset. We then feed the DCCs to our dashboard, where we contextualize them in the data map and also highlight the most similar data points in the dataset. By investigating similarities and differences between the data points, we were able to diagnose several spurious correlations, which we categorize into three different groups and a category of artifacts. Furthermore, we incorporate GPT-3 suggestions to

³We used the `roberta-large-wanli` model released on huggingface (Wolf et al., 2020).

allow for effective and diverse model-in-the-loop adversarial data generation. Therefore, our dashboard opens up future work on adversarial test suite generation and counterfactual augmentation.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive and thorough feedback as well as Frederic Boesel and Steven H. Wang for their contributions in the early stages of the project. We would also like to thank Anej Svete for his helpful comments on the final version of the paper. This work was funded by the ETH AI Center.

Ethics Statement

The authors foresee no ethical concerns with the research presented in this paper.

References

- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- 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.
- Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. [Entailment, intensionality and text understanding](#). In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 38–45.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. [The PASCAL recognising textual entailment challenge](#). In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Matt Gardner, William Merrill, Jesse Dodge, Matthew Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. 2021. [Competency problems: On finding and removing artifacts in language data](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1801–1813, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- William Huang, Haokun Liu, and Samuel R. Bowman. 2020. [Counterfactually-augmented SNLI training data does not yield better generalization than unaugmented data](#). In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 82–87, Online. Association for Computational Linguistics.
- Robin Jia and Percy Liang. 2017. [Adversarial examples for evaluating reading comprehension systems](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.
- Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020. [More bang for your buck: Natural perturbation for robust question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170, Online. Association for Computational Linguistics.
- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. [Dynabench: Rethinking benchmarking in NLP](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.
- Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. [WANLI: Worker and AI collaboration for natural language inference dataset creation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#).
- 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, Florence, Italy. Association for Computational Linguistics.
- Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. [Adversarial](#)

NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Thilo Spinner, Udo Schlegel, Hanna Schäfer, and Menatallah El-Assady. 2020. *explAIner: A visual analytics framework for interactive and explainable machine learning*. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1064–1074.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. 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.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. *Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models*. 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 6707–6723, Online. Association for Computational Linguistics.

LATEXSOLVER: a Hierarchical Semantic Parsing of LaTeX Document into Code for an Assistive Optimization Modeling Application

Rindra Ramamonjison¹, Timothy T. Yu¹, Linzi Xing^{1,2}, Mahdi Mostajabdaveh¹, Xiaorui Li¹, Xiaojin Fu³, Xiongwei Han³, Yuanzhe Chen³, Ren Li⁴, Kun Mao⁵, Yong Zhang¹

¹Huawei Technologies Canada, ²University of British Columbia

³Huawei Noah's Ark Lab, ⁴Huawei's UCD Center, ⁵Huawei Cloud Computing Technologies

{rindranirina.ramamonjison, timothy.t.yu, linzi.xing, mahdi.mostajabdaveh1, xiaorui.li, fuxiaojin, hanxiongwei, chenyanzhe, liren09, maokun, yong.zhang3}@huawei.com

Abstract

We demonstrate an interactive system to help operations research (OR) practitioners convert the mathematical formulation of optimization problems from LaTeX document format into the solver modeling language. In practice, a manual translation is cumbersome and time-consuming. Moreover, it requires an in-depth understanding of the problem description and a technical expertise to produce the modeling code. Thus, our proposed system LATEXSOLVER helps partially automate this conversion and help the users build optimization models more efficiently. In this paper, we describe its interface and the components of the hierarchical parsing system. A video demo walk-through is available online.¹

1 Introduction

Operations Research (OR) is a useful yet complex framework for optimal decision-making. For instance, OR has been used to increase bike-share ridership and efficiency (Beairsto et al., 2021), or to optimize wastewater collection and treatment (Tao et al., 2020) in cities. Despite its importance in many fields, the OR process is both time-consuming and knowledge-intensive. First, the problem specifications provided by domain experts must be formulated in mathematical form (Carter and Price, 2017). Then, the formulation needs to be converted into a model code that optimization solvers can interpret. Next, data parameters must be collected and used to instantiate the optimization model. Finally, a proper solver needs to be selected to solve the given problem and find an optimal solution. Traditionally, the domain expert hires an OR expert to handle these strenuous tasks and build the right model for the problem of interest.

There are two shortcomings in the above process, which increases the project cost and duration.

1. First, the formulation needs to be written twice. OR experts typically write the problem formulation as a LaTeX document containing both natural language (NL) descriptions and math formulas. Then, they translate it into code using a solver-specific modeling language. This manual work creates a bottleneck and a mental overhead for the OR experts.
2. Second, the optimization models are saved in two different formats namely the LaTeX document format and the modeling code format. This makes it difficult to manage, edit or share the optimization models. Even if software versioning systems can be used to track the document or code changes, they are quite limited and cumbersome for this purpose.

To address these shortcomings, we introduce LATEXSOLVER, an interactive system that takes as input the problem description of a model in LaTeX and partially automates its translation into modeling code. To the best of our knowledge, this is the first modeling tool that accepts an unstructured and multi-modal LaTeX document as input format.

Moreover, we introduce a unified *symbolic model* representation, which decouples the translation procedure into two stages. First, our system combines information extraction methods (i.e., text segmentation, entity recognition, relation extraction) with grammar-based parsing to extract the symbolic model. In the second stage, our system uses the actual data parameters to instantiate the symbolic model and generate the modeling code.

Finally, our intuitive user interface displays the symbolic model as a graph of the model elements as shown in Figure 1. Each node shows the formula and metadata of an element and allows the user to review and edit it before it is turned into modeling code. This added flexibility puts the user in control.

¹<https://bit.ly/3kuOm3x>

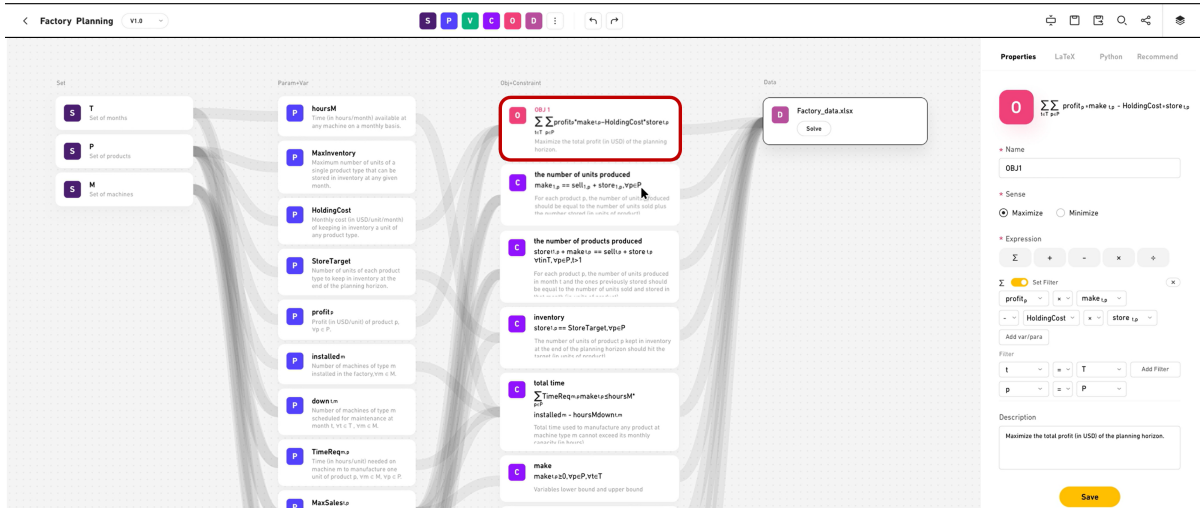


Figure 1: Screenshot of the LATEXSOLVER interactive application. The system displays the parsed symbolic model graphically where the user can interact with the cards representing different modeling components. The right panel empowers the user to revise each component efficiently and with ease.

2 Related Work

Modeling aid tools for operations research.

Past efforts to improve the experience of model builders has centered around creating better modeling languages. While they can enable rapid prototyping (Koch, 2004), the modeler still needs to learn these languages and manually convert the mathematical formulation into an equivalent code.

To alleviate the technical barriers and make solvers more accessible, alternative input formats have also been proposed such as Excel spreadsheets (Lin and Schrage, 2009), web forms (Triantafyllidis and Papageorgiou, 2018; Esche et al., 2017), and natural language (IBM, 2022). In comparison, our system is the first to accept a multi-modal markup document that contains natural language descriptions as well as mathematical equations.

Information extraction from scientific documents.

Recently, information extraction from scientific documents has received increasing research interests. In fact, scientific documents are different from natural language texts due to the syntactic difference and to the presence of symbols and formulas (Lai et al., 2022). (Beltagy et al., 2019) proposed SciBERT is an example of a pre-trained language model on corpora for different scientific domains. SciBERT was used in Lee and Na (2022) and Popovic et al. (2022) as the backbone for entity recognition and relation extraction tasks. Moreover, Lee and Na (2022) reframed entity recognition and relation extraction tasks as machine reading comprehension to leverage the mathematical knowl-

edge learned by SciBERT. For our system, we also adopt SciBERT and fine-tune it using our labeled dataset for the information extraction tasks. In addition, we use a neural text segmentation and labeling model as the first step to divide the input markup document into declaration segments.

Program synthesis and math word problems.

Language models pretrained on source code have shown some promising performance in generating or summarizing computer programs (et al., 2021). These models have also been used to generate programs that solve math problems (Drori et al., 2022). Nonetheless, recent studies have shown that even the largest models such as Codex can still hallucinate and produce erroneous code (Xu et al., 2022). Direct translation from an unstructured markup language to code is an under-explored task and current techniques does not deliver consistently accurate results. Instead, we divide it into smaller tasks and simplify the parsing by leveraging grammar-based parsers.

3 System Overview

Figure 1 showcases the graphical user interface of our LATEXSOLVER web application built using the Vue frontend framework (Vue.js, 2014).

This interface enables users to upload an optimization problem description in LaTeX, composed of both natural language and mathematical formulas. After going through the parsing process, the input LaTeX document will be transformed into a symbolic model that serves as a united repre-

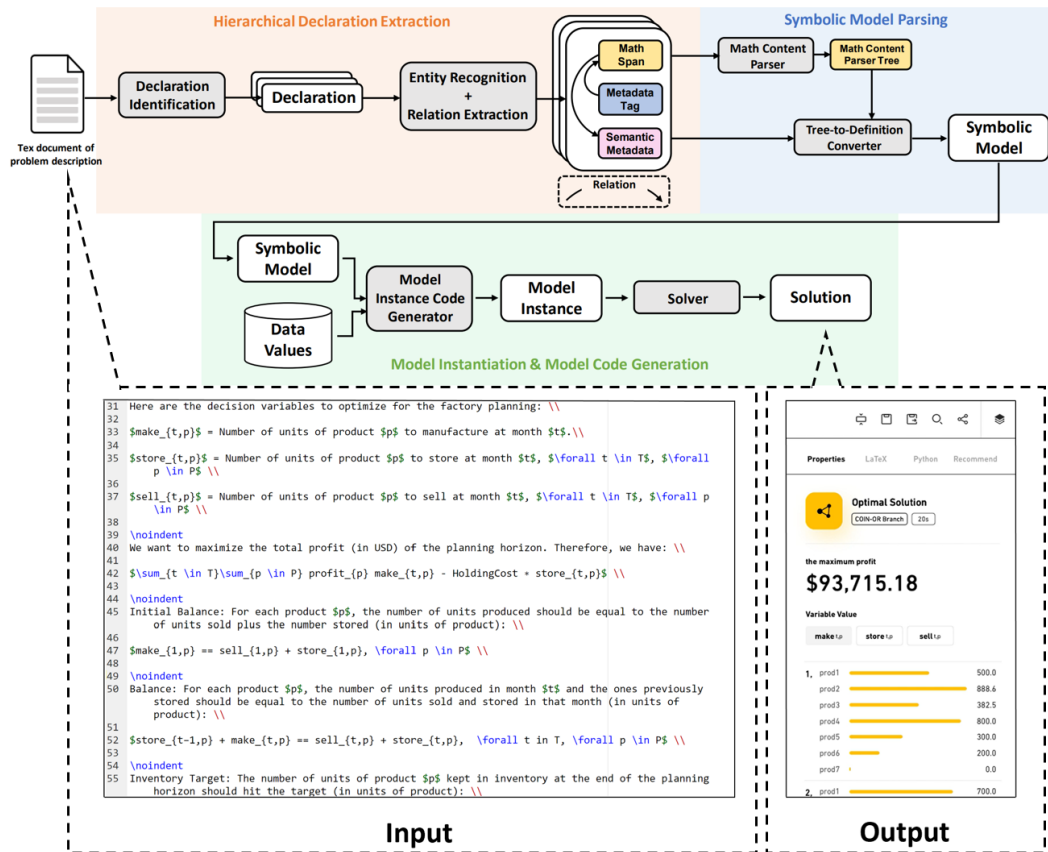


Figure 2: System diagram for LATEXSOLVER.

sentation of the input optimization problem. As an example shown in the main panel of Figure 1, a symbolic model comprises model elements extracted from the document and each element is categorized as one of $\{Set, Parameter, Objective, Variable, Constraint, Objective\}$ (listed on top of the interface). All parameters and variables in the symbolic model are represented by symbols rather than actual values. Our user interface is designed to display the extracted symbolic model as a graph, in which each node contains the formula and metadata of a model element and dynamic links are used to highlight the relationships between symbolic elements. To enhance user engagement and system flexibility, we allow users to review and edit the displayed model elements. Users can click on each model element card to reveal its detailed properties in the right-side panel of the interface, where they can edit the properties as desired. Furthermore, users have the ability to add or delete elements as needed. Once the user is satisfied with the updated symbolic model, our system will guide the user to upload data values to instantiate the symbolic model and generate the corresponding model code instance for the optimization solver to compute for

solutions.

The backend workflow of LATEXSOLVER is illustrated in Figure 2. The system’s pipeline consists of three major stages, namely *hierarchical declaration extraction*, *symbolic model parsing* and *model code generation*. Given a problem description in LaTeX as input, the system first segments it into a set of declarations, each of which describes a specific model element and includes declaration entities linked by corresponding relations. This stage employs three neural models that were initially introduced for three NLP tasks: text segmentation and labeling, entity recognition, and relation extraction respectively. In the subsequent stage, the extracted declarations are transformed into a symbolic optimization model using a context-free grammar-based parser, which leverages the metadata obtained in the previous stage as supplementary information. The resulting symbolic model is then passed into a solver API-specified model code instance generator, together with user-specified data values, to generate the model code that is ready to be processed by optimization solvers.

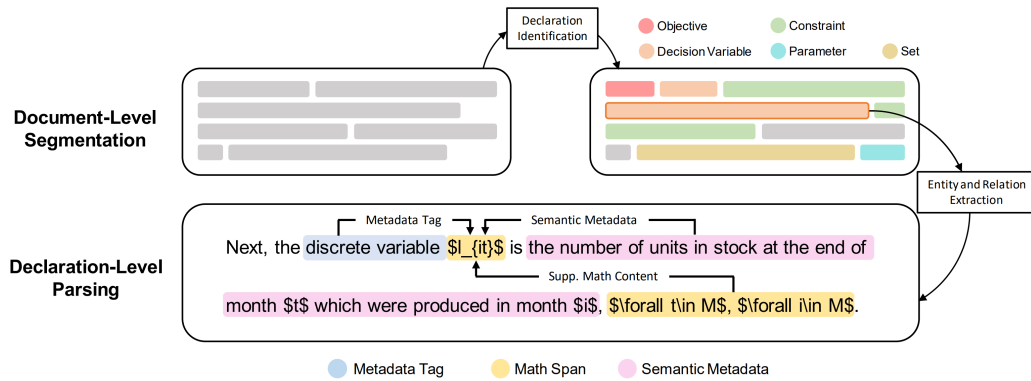


Figure 3: The illustration of Hierarchical Declaration Extraction.

4 Hierarchical Declaration Extraction

A declaration in this paper is, by definition, a multi-modal segment consisting of one or more sentences written in a mixture of text and math content, as exemplified in Figure 3. Each declaration typically describes one particular model element, such as an *objective*, *constraint*, *decision variable*, *parameter* or *set*. As the first stage of LATEXSOLVER, the system takes a LaTeX document as input and extracts declarations from it in a hierarchical manner. Specifically, the system first performs document-level declaration identification to extract and label all the text segments in the document, each associated with one declaration (§4.1). Next, our system performs entity recognition and entity linking within each extracted declaration segment (§4.2).

4.1 Document-Level Declaration Identification

In order to identify all declarations contained in the document, as well as assign each identified declaration a label indicating the model element it contributes to, we propose to re-purpose a neural model originally proposed in Barrow et al. (2020) for text segmentation and labeling. In this work, we employ this model for declaration segmentation and labeling.

Figure 4 illustrates the high-level architecture of the neural model we used for declaration segmentation and labeling. In practice, the system accepts the input document formalized as a sequence of consecutive sentences². However, the structure of a LaTeX document is usually not flat but contains nested content blocks organized as bullet lists, or covered in captions of figures and tables. Therefore,

²We applied `nltk.sent_tokenize` for sentence segmentation.

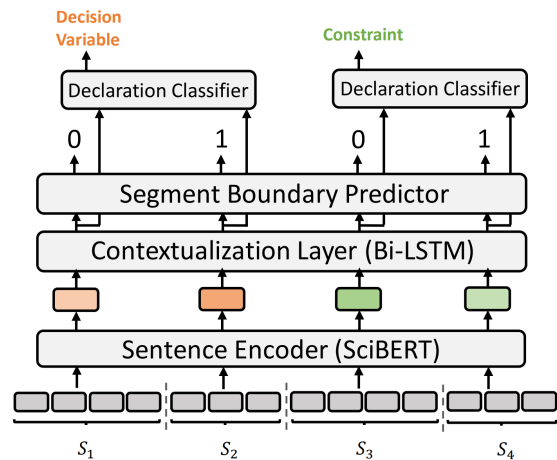


Figure 4: The model architecture for declaration segmentation and labeling.

we initially carry out a rule-based pre-processing step to detect and eliminate these structures by converting the content in the nested form to the flat form before passing them into the model. As the first layer of the neural model, a sentence encoder is in place to yield low-level features (embeddings) for the input sentences. Taking into consideration that (1) documents in LaTeX are more likely in the scientific domain, and (2) sentences within these documents are likely to have both text and mathematical content, we choose SciBERT (Beltagy et al., 2019) as our sentence encoder, which is equipped with a rule-based symbol tokenizer proposed in Lee and Na (2022) to alleviate the limitation of SciBERT’s tokenizer in detecting the boundaries of mathematical symbols.

Given the sentence embeddings obtained from the SciBERT sentence encoder, a document-level contextualization layer (Bi-LSTM) returns an ordered set of sentence hidden states for two objectives: (1) declaration segment boundary prediction

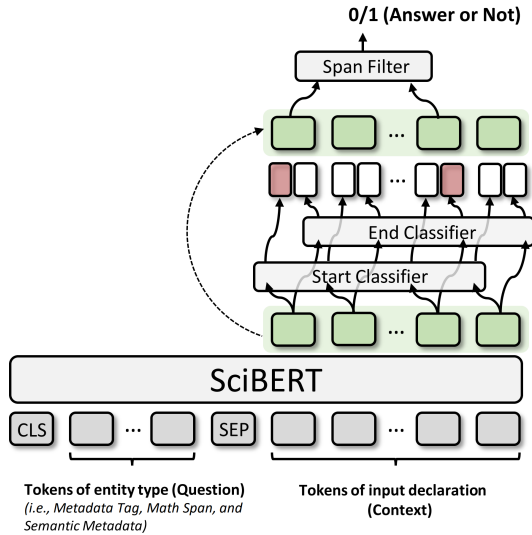


Figure 5: The model architecture for entity recognition.

and (2) declaration labeling. Concretely, a multi-layer perceptron (MLP) followed by softmax activation serves as a binary boundary predictor, where label "1" means the corresponding sentence is the end of a segment, and "0" otherwise. After segmenting the document, we further apply mean pooling over the sentence hidden states within each predicted segment to predict the declaration label for the segment. Another MLP+softmax layer is utilized to classify each declaration segment into one of the following classes: $\{Objective, Constraint, Decision Variable, Parameter, Set, Others\}$. This framework is optimized by minimizing the cross-entropy losses for both objectives of declaration segmentation and labeling.

4.2 Entity Recognition and Relation Extraction

Once the underlying declarations have been extracted from the input LaTeX document, the next step entails extracting entities within each declaration, as well as the relations linking up these entities. To achieve this, we leverage the entity recognition and relation extraction models proposed in Lee and Na (2022), initially devised for machine reading comprehension on documents containing mathematical symbols.

The entity recognition model formulates the process of extracting entities as a machine reading comprehension task by providing an entity type as a question and utilizing SciBERT as the backbone to extract mentions of this entity type in a declaration segment as answers. As shown in Figure 5, a

given input declaration is deemed as context and concatenated with each of the three pre-defined entities types (i.e., *Metadata Tag*, *Math Span*, *Semantic Metadata*) in our labeled corpus. Subsequently, the Question+Context concatenation is passed into SciBERT, and the output hidden representations of the tokens covered by the declaration (context) are used to estimate the probability of each token being the start or end (i.e., i_{start} or i_{end}) of a mention (answer) of the concatenated entity type (question). Next, for any span of (i_{start}, i_{end}) , another binary classifier is applied to predict whether the span is the answer to extract.

Similar to the entity recognition model described above, we also leverage SciBERT to perform relation extraction between pairs of entities within each declaration segment by simply encoding entities and doing relation prediction based on the concatenation of entity representations (mean of token embeddings covered in the text span of entities) obtained from SciBERT (Lee and Na, 2022). We pre-define four types of relations, namely *Metadata Tag*, *Semantic Metadata* and *Supplementary Math Content* and *NIL*, where *NIL* indicates that no relation exists between the two entities.

For both entity recognition and relation extraction, we set some heuristic rules to refine models' predictions by cleaning up the entities which are unlikely to co-occur within a declaration, as well as relations with the type unlikely to appear to link two certain entities.

5 Symbolic Model Parsing and Model Code Generation

This section describes the process of converting the detected entities and their relations into a symbolic model which is eventually generated into modeling code, as shown in Figure 6. The symbolic model and concrete data values, such as sets and parameter values, are passed to the model instance code generator which converts them into code based on the target modeling language or solver API. The generated code consists of the model components and their corresponding data values. By decoupling the symbolic model and data values, we allow the users to evaluate their model and problem more efficiently. They can modify the model to evaluate variants of the model or easily change data values to examine different scenarios of the problem.

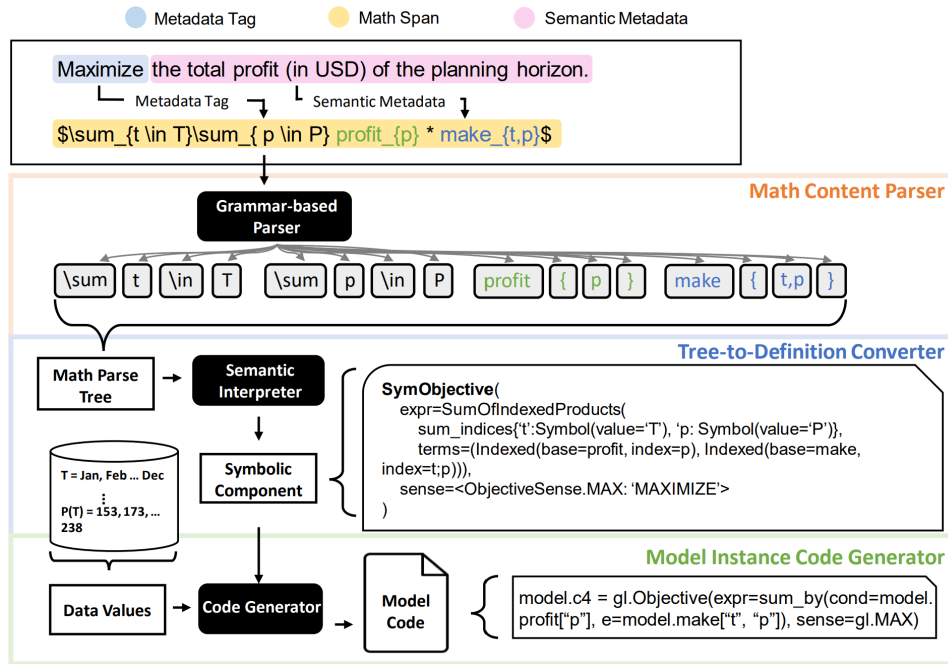


Figure 6: The illustration of symbolic model parsing and model code generation process.

5.1 Symbolic Model Parsing

As part of LATEXSOLVER, we have implemented a symbolic model, which is an intermediate representation of the problem that lies between the natural language & math problem description and model code. This symbolic representation of the model maps the math formulas into a nested structure of symbolic elements before data instantiation. The symbolic model parsing is performed by two modules: (1) a math content parser, and (2) a tree-to-definition converter. An example of both modules is shown in Figure 6.

First, the math content parser converts the math content spans of each model component declaration detected by the declaration extraction stage into a parse tree. To do so, the LaTeX math formulas are parsed using the grammar-based parser generator ANTLR4 (Parr, 2013). ANTLR4 detects the required attributes of modeling components. Specifically, indexed parameters and variables are parsed along with the set that the indices are bound to. The parser also parses constraint math formulas into the constraint expression, comparison operator ($<$, $<=$, $>$, $>=$, $==$), and the right-hand-side equation. Finally, the objective math formula would contain the objective sense and the objective function. These math formulas are processed into a parse tree where the leaves contain the important fields required to populate the symbolic model. Figure 6 shows a graphical representation of the

output parse tree of the grammar-based parser.

Next, the tree-to-definition converter loops through the parse tree, it processes the detected model components, and creates a new model component in the symbolic model class. As these elements are detected, they populate the attributes of the corresponding components in the symbolic model class in a nested manner.

5.2 Model Code Generation

The model instance code generator leverages Python's metaprogramming mechanism to generate the modeling code during runtime. The pipeline begins with a template code string that imports the required Python libraries for the solver API. The model code generator adds to the model based on the specifications of the API. In our implementation, we used the Huawei OptVerse solver and its Grassland API (Li et al., 2021). The model instance code generator reads in the data values from spreadsheet files using predefined sheet references and adds the data initialization commands in the generated code. Then, it will loop through the elements of the symbolic model to translate it into the solver API code to declare the sets, variables, objective and constraints.

6 Limitations and Future Works

The LATEXSOLVER system is a useful tool that partially automates the conversion of a problem

description into an optimization model, but some limitations highlight potential areas for improvement. First, it was assumed that each declaration contains all complete information that needs to be parsed into a model element. While our proposed system can handle unstructured text at the declaration-level, it is a more challenging problem if information about one model component is scattered across declarations. We believe that in most cases, the input documents would satisfy this assumption. Second, this system requires the LaTeX document input to be accurate and errors may cause issues that cascade through the steps of the system (e.g., an error in segmenting and labeling the declarations may yield errors in the entity recognition and relation extraction tasks). The interactive application addresses this by keeping the human in the loop allowing the user to interact with the detected components and their relations. Finally, with the rapid advancements in LLMs (Li et al., 2023; OpenAI, 2023), domain-specific tools such as LATEXSOLVER could be used in conjunction with general-purpose language models.

7 Conclusion

In this paper, we introduce LATEXSOLVER, an interactive system to help operations research practitioners efficiently convert the mathematical formulation of optimization problems from the unstructured and multi-modal LaTeX document format into the solver modeling language. The system follows a two-step process of converting the LaTeX document to a symbolic model and then to model code. Users can easily review and edit the symbolic model automatically extracted from the LaTeX document with an intuitive interface to ensure the system's reliability.

Ethics Statement

The LATEXSOLVER system presented in this paper aims to partially automate the process of converting problem descriptions in LaTeX to model codes, thereby helping OR experts build optimization models more efficiently. As the system's input and output are transparent to users and users can control the model-building procedure by interacting with the system, the harm to users resulting from the errors produced by the system is limited. However, our system may be used in certain circumstances considered sensitive or critical, such as power grid or flights scheduling. In such cases, the

system should be used with caution and the modeling process should be investigated by domain experts. Additionally, given the historic application of operations research in tactical military operations, it is critical to understand the potential negative impact of misapplying this technology to society. Therefore, users of our system must be aware of any ethical concern come with military applications of this technology.

References

- Joe Barrow, Rajiv Jain, Vlad Morariu, Varun Manjunatha, Douglas Oard, and Philip Resnik. 2020. [A joint model for document segmentation and segment labeling](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 313–322, 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.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Michael W Carter and Camille C Price. 2017. *Operations research: a practical introduction*. Crc Press.
- Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, Roman Wang, Nikhil Singh, Taylor L. Patti, Jayson Lynch, Avi Shporer, Nakul Verma, Eugene Wu, and Gilbert Strang. 2022. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences (PNAS)*, 119(32).
- Erik Esche, Christian Hoffmann, Markus Illner, David Müller, Sandra Fillinger, Gregor Tolksdorf, Henning Bonart, Günter Wozny, and Jens-Uwe Repke. 2017. Mosaic—enabling large-scale equation-based flow sheet optimization. *Chemie Ingenieur Technik*, 89(5):620–635.
- Chen et al. 2021. [Evaluating large language models trained on code](#).
- IBM. 2022. [Modeling assistant models \(decision optimization\)](#).

- Thorsten Koch. 2004. *Rapid Mathematical Prototyping*. Ph.D. thesis, Technische Universität Berlin.
- Viet Lai, Amir Pouran Ben Veyseh, Franck Dernoncourt, and Thien Nguyen. 2022. [SemEval 2022 task 12: Symlink - linking mathematical symbols to their descriptions](#). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1671–1678, Seattle, United States. Association for Computational Linguistics.
- Sung-Min Lee and Seung-Hoon Na. 2022. [JBNU-CCLab at SemEval-2022 task 12: Machine reading comprehension and span pair classification for linking mathematical symbols to their descriptions](#). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1679–1686, Seattle, United States. Association for Computational Linguistics.
- Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliachko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvasi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. [Starcoder: may the source be with you!](#)
- Xihan Li, Xiongwei Han, Zhishuo Zhou, Mingxuan Yuan, Jia Zeng, and Jun Wang. 2021. [Grassland: A rapid algebraic modeling system for million-variable optimization](#).
- Youdong Lin and Linus Schrage. 2009. The global solver in the lindo api. *Optimization Methods & Software*, 24(4-5):657–668.
- OpenAI. 2023. [Gpt-4 technical report](#).
- Terence Parr. 2013. The definitive antlr 4 reference. *The Definitive ANTLR 4 Reference*, pages 1–326.
- Nicholas Popovic, Walter Laurito, and Michael Färber. 2022. [AIFB-WebScience at SemEval-2022 task 12: Relation extraction first - using relation extraction to identify entities](#). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1687–1694, Seattle, United States. Association for Computational Linguistics.
- 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.
- Charalampos P Triantafyllidis and Lazaros G Papa-georgiou. 2018. An integrated platform for intuitive mathematical programming modeling using latex. *PeerJ Computer Science*, 4:e161.
- Vue.js. 2014. Vue.js – the progressive javascript framework. <https://vuejs.org/>. Accessed: 2023-05-20.
- Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, MAPS 2022*, page 1–10, New York, NY, USA. Association for Computing Machinery.

Alfred: A System for Prompted Weak Supervision

Peilin Yu Stephen H. Bach

Department of Computer Science

Brown University

{peilin_yu, stephen_bach}@brown.edu

Abstract

Alfred is the first system for programmatic weak supervision (PWS) that creates training data for machine learning by prompting. In contrast to typical PWS systems where weak supervision sources are programs coded by experts, Alfred enables users to encode their subject matter expertise via natural language prompts for language and vision-language models. Alfred provides a simple Python interface for the key steps of this emerging paradigm, with a high-throughput backend for large-scale data labeling. Users can quickly create, evaluate, and refine their prompt-based weak supervision sources; map the results to weak labels; and resolve their disagreements with a label model. Alfred enables a seamless local development experience backed by models served from self-managed computing clusters. It automatically optimizes the execution of prompts with optimized batching mechanisms. We find that this optimization improves query throughput by $2.9\times$ versus a naive approach. We present two example use cases demonstrating Alfred on YouTube comment spam detection and pet breeds classification. Alfred is open source, available at <https://github.com/BatsResearch/alfred>.

1 Introduction

Acquiring labeled data is a significant challenge for machine learning for its time-consuming and expensive nature. Programmatic weak supervision (PWS) provides a more efficient method of data annotation by using noisy heuristics to label data. In a typical PWS setup, domain experts design labeling functions (LFs) as programs that vote for a label or abstain. (Ratner et al., 2016, 2017) Recently, there has been a growing interest in creating LFs from large, pre-trained models through prompting (Smith et al., 2022; Arora et al., 2022; Zhang et al., 2022b). In the shift to this new setting, executing LFs goes from the least to the most computationally expensive part of the process, highlighting

the importance of providing a software infrastructure that facilitates efficient development. However, existing toolkits for large language models mainly prioritize prompt templating and tuning, leaving an unmet need for a system that connects prompting with the creation of training data.

Prompted models offer a unique opportunity to enhance existing PWS systems. Traditional PWS systems require programming LFs with code that specifies heuristic domain knowledge. With large pre-trained models, natural language-based prompts can be used as LFs, also known as prompted LFs (Smith et al., 2022). This approach allows the easy expression of complex rules that were previously difficult to specify using code, as the example in Figure 1 shows. The ability to use prompts to define labeling functions simplifies and streamlines the weak supervision process, as well as potentially elevating the quality of the annotations. This benefit is particularly helpful for tasks involving computer vision, where previously PWS has been limited to tasks for which models can identify key features (such as objects) over which to program rules. Whether the domain is language or multi-modal, prompts let users experiment with different heuristics (and phrasings of those heuristics). Therefore, enabling an iterative development experience is essential for the success of a prompt-based PWS system.

The switch to prompted models for weak supervision also presents significant challenges. It first requires rethinking the abstractions and workflow of first-generation PWS systems. Instead of editing code and managing libraries of functions, users must manage libraries of prompts, track their outputs on multiple datasets, and develop strategies for mapping those outputs to labels. This change is complicated by large models' demand for computational resources. The throughput of the models is a new development bottleneck. The extra overhead of remotely hosted models is a further hindrance to



Figure 1: Examples of a labeling function versus a prompted labeling function. For the first example, each expresses supervision relating mentions of President Biden to the category of politics. Instead of specifying an intricate regular expression, a prompted labeling function uses the prompt “Text: [[instance]] Does this text mention President Biden?” where [[instance]] is replaced by the news article to be labeled. The response is mapped to a vote on the true label. The second example demonstrates how heuristics about positive sentiment that were previously hard to define can be flexibly expressed as a natural language question. Instead of defining a set of keywords for fuzzy sentiment matching, we can simply ask large pretrained models for answers about the sentiment. For the third example, we consider an animal labeling task where we use the visual attributes “stripes” to vote for the classes TIGER and ZEBRA. Previously, we would need to first collect supervised training data for attributes like stripes and then train classifiers to make the decisions. With modern vision-language models (e.g. CLIP), we can simply express the attribute detection task as a set of candidate prompts.

the iterative workflow of weak supervision.

Existing open-source software for prompting concentrates on prompt engineering (Orr, 2022; Bach et al., 2022), prompt chains (Chase, 2022) or continuous (i.e., soft) prompt tuning (Ding et al., 2022); placing less emphasis on throughput for a large-model-in-the-loop workflow. Additionally, many existing open-source systems have not developed support for vision-language models, despite their benefits for data labeling. Incorporating large pre-trained models into a PWS system remains an open problem in the open-source software space, requiring innovative solutions to address these challenges and complement existing software focused on other aspects of prompting.

We present Alfred, a versatile PWS system that leverages large pre-trained models for image and text annotations. Alfred aims to provide an environment for the rapid development of prompt-based supervision, while maintaining a consistent development experience similar to established PWS systems. We designed Alfred with usability and efficiency in mind, aiming to provide a rapid and

smooth experience for developing prompt-based supervision. Alfred supports popular large language models from Hugging Face’s transformer package (Wolf et al., 2020), including the GPT family (Radford et al., 2019), the T5 family (Raffel et al., 2020), etc., and vision-language models like CLIP (Radford et al., 2021), etc. Alfred also supports local ONNX models, or API-based models from OpenAI, AI21, and Cohere. Moreover, Alfred provides easy templating tools to help users quickly create, evaluate, and refine prompted LFs. Alfred offers easy ways to deploy inference servers remotely, in addition to local model hosting. Alfred also optimizes model inference throughput with improved batching techniques and provides utilities for efficient LLM deployment and interaction. Finally, Alfred contains a library of label models to distill the outputs of prompted labeling functions into the final training datasets for downstream end models.

Alfred is a prototype for a second generation of PWS systems with prompting at their core. To this end, we highlight three key feature of Alfred:

- **Prompt-based weak supervision for images**

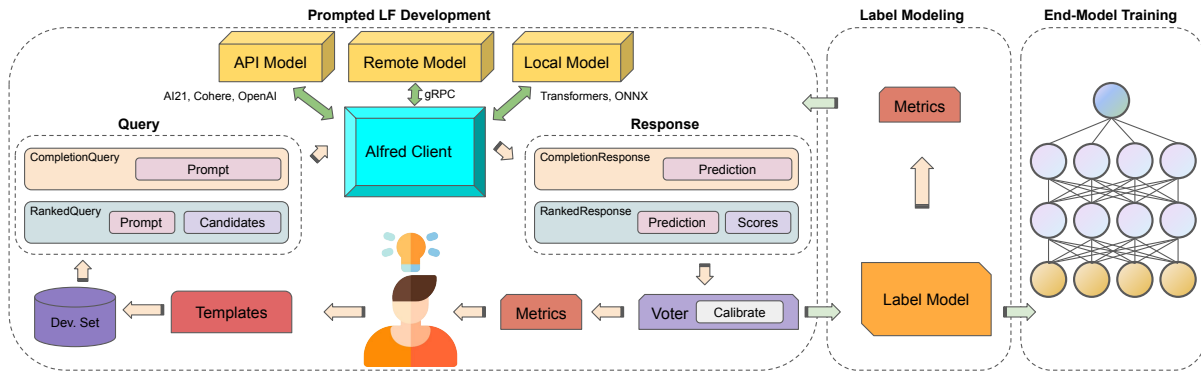


Figure 2: A typical workflow for programmatic weak supervision with Alfred. First, developers use Alfred to iteratively design, evaluate, and refine their prompted labeling functions (LFs). They use prompt Templates to generate Queries to Models based on data. The responses of Models are mapped to votes on the true labels for examples by Voters, which can be calibrated. Then the included Label Models combine the noisy votes to produce probabilistic training labels for an end model.

and text. Alfred provides the necessary tools for users to create, evaluate, and refine prompt templates for both image and text weak supervision tasks. The inclusion of a query caching system that automatically stores and updates model responses facilitates development.

- **Optimized inference throughput.** Alfred implements a dynamic batching mechanism that optimizes large sets of prompts. This feature allows models hosted by Alfred to achieve 2-3× greater throughput than naive implementations.
- **Seamless local development experience.** Alfred can host models remotely and make them accessible to developers via gRPC, a high-throughput protocol for remote procedure calls.¹ It enables sending datasets to servers in large chunks to maintain higher throughput. Alfred also implements a SSH-based port-forwarding utility for the gRPC connection, easing deployment on shared clusters such as those at universities.

2 Related Work and Background

Alfred sits at the intersection of programmatic weak supervision and large pretrained models. In this section, we overview the most related work.

Programmatic Weak Supervision. Traditionally, supervised learning relied on manually labeled data, and data labeling has been seen as a key bottleneck for many applications. Recently, a family of programmatic weak supervision (PWS) methods have offered an alternative to costly manual annotations by incorporating multiple sources of noisy labels to create training datasets (Zhang et al., 2022a).

¹grpc.io

Typically, a PWS system such as Snorkel (Ratner et al., 2017) has a three-stage setup: First, developers will create heuristics called labeling functions (LFs) that vote on the label for an example (or abstain). Second, a label model will reconcile the noisy votes and provide probabilistic labels for the data. Finally, freshly annotated data is used to train an end model with a noise-aware loss objective (e.g. in a classification setting, this can be a soft cross entropy (Ratner et al., 2016)). Alfred focuses on the first two stages and aim to efficiently incorporate modern large pretrained models into the development workflow.

Prompting for Pre-Trained Models. With the emergence of large pre-trained language and vision-language models, prompting has become a popular approach to many few-shot and zero-shot tasks (Brown et al., 2020; Schick and Schütze, 2021a; Radford et al., 2021). Prompting can create training examples, generate data, modify datasets, and improve model reasoning (Schick and Schütze, 2021b; Ye et al., 2022; Chia et al., 2022; Wu et al., 2022; Wang et al., 2022; Wei et al., 2022; Zelikman et al., 2022). This presents a unique opportunity for combining prompting for large pretrained models and weak supervision. Recent studies have investigated strategies to combine large language models into weak supervision frameworks (Smith et al., 2022; Chen et al., 2022; Arora et al., 2022; Zhang et al., 2022b). Alfred aims to provide a platform for the rapid development of weak supervision applications that rely on large pre-trained language and vision-language models, as well as enable experimentation with new ways of using those models.

Systems for Prompt Development. Prompting has led to the development of software toolkits that aid in prompting and research across various tasks. Many tools have been developed for various use cases with large language models. PromptSource (Bach et al., 2022) is a development environment for creating and archiving sets of prompts. OpenPrompt (Ding et al., 2022) is a library focused on tuning prompts and prompted models with training data. Manifest (Orr, 2022) provides a unified front end for prompting large language models via different APIs. LangChain (Chase, 2022) provides convenient utilities for building applications that chain together multiple prompts and outputs. To complement the existing tools in this growing space, Alfred is designed to be a PWS system based on prompting both large language and vision-language models.

3 Prompted LF Development

Alfred is designed to enable the development and application of prompted labeling functions (LFs). Compared to the typical workflow of PWS systems, where developing LFs is not computationally demanding, developing prompted LFs has model inference as a bottleneck. These large models are often hosted remotely on virtual instances or computing clusters, which can add to the challenge of iterative prompt development. In an iterative development environment, creating prompted labeling functions requires a platform that provides optimal throughput and low latency for a rapid local development experience. To illustrate Alfred’s key focuses, we illustrate a typical workflow (Figure 2) for using Alfred to create a training dataset:

Step 1: Task Setup For a large model to be used in the development loop, developers can elect to use either self-hosted models or API-based models. For self-hosted models, Alfred provides an `AlfredServer` to host the model on cloud or cluster nodes. As the main development interface, the user can simply start a `Client` by specifying the type of model. Before creating prompted LFs, users need to familiarize themselves with the task by exploring the raw, unlabeled dataset. If there is no development subset available, the developer can annotate a small portion of the data as a held-out evaluation benchmark. Alfred implements a `Dataset` class based on Apache Arrow for fast data access. User may load a `Dataset` from CSV, JSON or Dataframe objects. It also offers direct support

for datasets from the Hugging Face ‘Dataset’ package (Lhoest et al., 2021). During the exploration process, developers may gain insights into the data and the label space, and identify potentially useful heuristics. Moreover, users can freely experiment with prompts with a few unlabeled instances by directly interacting with the `Client`.

Step 2: Iterative Prompt Development: When the user is ready for prompt development, they can use a `Template` to define a prompted LF for either text completion or scoring schemes. A `Template`, given a `Dataset`, will produce the corresponding `Query` objects. `Client` will return the appropriate `Response` object for each `Query`. To map the model responses to votes, users define the corresponding `Voter` and identify the label maps and matching functions to be used for each prompted LFs. Label maps define how potential model responses are associated with the label space. Matching functions specify how the `Voter` determines a match. By default, Alfred employs an exact match mechanism, but this can be substituted with user-defined matching functions for uncased matching or embedding similarity matching, etc. A `Voter` can be optionally calibrated to reduce model-specific biases (Zhao et al., 2021). With the model responses and the `Voter`, users can obtain the label votes for each of their prompted LFs and examples in a matrix format. Finally, users can evaluate the quality of their prompted LFs using a set-aside development `Dataset` with desired metrics. Here, users can continue to refine their prompted LFs and iterate as necessary. Once users are satisfied with the performance of their development benchmark, they may proceed.

Step 3: Aggregate Prompt Responses Finally, Alfred can aggregate the votes from each `Voter` with a `LabelModel` to produce probabilistic estimates of the true labels for the examples. Alfred also supports *partial labels*, i.e., labels that narrow down the possible set of classes but are not specific enough to vote on a single class (Yu et al., 2022). The probabilistic labels can then be used to train a wide range of end models.

4 System Design

In this section, we describe and highlight the key design decisions for Alfred.

4.1 Query and Response Types

We identify two main patterns using prompts

```

from alfred import Client
from alfred.fm.query import RankedQuery,
                             CompletionQuery
LMClient = Client(...)

headline = "Liverpool wins 7-0!"

LMClient(
    CompletionQuery(headline
        + " What is the topic of this headline?")
)
# Example Response:
# >> CompletionResponse(prediction="Football")

LMClient(
    RankedQuery(headline
        + " What is the topic of this headline?",
        candidate=["Sports", "Politics",
                  "Tech", "Business"])
)
# Example Response:
# >> RankedResponse(prediction="Sports",
#     scores={"Sports":0.76, "Politics":0.10,
#            "Tech":0.07, "Business":0.07 })

```

Figure 3: Typed `Query` and `Response` in Alfred

for PWS: text completion and scoring. Text completion is when a language model generates responses using a heuristic decoding strategy over the whole model vocabulary, while scoring is when a language ranks candidate completions or a vision-language model ranks candidate prompts, i.e., captions. Alfred implements typed `Query` and `Response` classes for these two patterns (Figure 3). Upon applying the `Template` operation on a dataset instance, it produces either a `CompletionQuery` or a `RankedQuery` for each instance based on the `Template` definition. The resulting query can be directly fed into the `Client`. The `Client` then returns a corresponding `CompletionResponse` or `RankedResponse` with the prediction as the main payload, along with any other requested or useful information, such as the logits for each candidate.

4.2 Templates for Prompted LFs

Prompt templates are at the core of systems for prompting. In Alfred, prompt templates are expressed as `Template` objects. For natural language tasks, users use the `StringTemplate` class. To produce `Query` objects, users can call ‘`Template.apply(instance)`’ or ‘`Template.apply_to_dataset(dataset)`.’ A `StringTemplate` is defined with a template string

with keywords enclosed by double square brackets, e.g. “[`text`] Does the previous context express spouse relation between [`entity_a`] and [`entity_b`]?”. An optional field for `Template` is ‘`answer_choices`,’ where one may specify the candidate completions. By specifying the ‘`answer_choices`,’ the `StringTemplate` would yield a `RankedQuery`. An example code snippet showing the creation of a `RankedQuery` is in Figure 4. For image annotation tasks, users may define an `ImageTemplate` by specifying the candidate prompts. Upon applying to images, `ImageTemplate` will produce `RankedQuery` objects with the images and candidate prompts.

4.3 Throughput Optimization

Alfred is designed to handle large numbers of queries. Self-hosted models from the Transformers package (Wolf et al., 2020) are set up to use model parallelization enabled by Accelerate (Sylvain Gugger, 2022), with user-customizable device maps for parallelizing the model across multiple GPUs. Alfred adopts a dynamic batching strategy that groups instances with similar lengths together and adjusts the input batch size dynamically to maximize model inference throughput. The core idea of the dynamic batching strategy is to group input instances with similar token lengths to minimize padding and maximize memory utilization.

With the dynamic batching strategy, on a node with 8 NVIDIA Tesla V100s, Alfred achieves a speedup of up to 2.5× and a token throughput increase of 2.9× for approximately 500 queries (~21,000 tokens) compared to an unoptimized strategy with T0++ (Sanh et al., 2022), an 11-billion parameter T5-based (Raffel et al., 2020) language model in FP32. Additionally, Alfred includes a client-side query caching system that automatically stores and updates model responses to facilitate prompt development and avoid redundant queries during development. Alfred also implements a server-side caching system for large multi-modal pretrained models such as CLIP. At inference time, Alfred will cache the input data with its corresponding encoded latent representations from different encoder head for each modalities. This server-side caching system effectively avoids redundant encoding computation on the server end.

```

from alfred.template import StringTemplate, ImageTemplate

string_template = StringTemplate(
    "Context: [[text]]\n\nIs the above text about weather?", answer_choices = ["Yes", "No"]
)
example = {'text': "A pleasant day with a sunny sky."}
prompt = string_template.apply(example)
# >> RankedQuery("Context: A pleasant day with a sunny sky.\n\nIs the above text about weather?",
#                 candidates=["Yes", "No"])

image_template = ImageTemplate(
    {"label": ["cat", "dog"]},
    template = "A photo of [[label]]."
)
example = cat_image
prompt = image_template.apply(example)
# >> RankedQuery(example, candidates=["A photo of cat.", "A photo of dog."])

```

Figure 4: Example code snippet for creating a `RankedQuery` from a `StringTemplate` or a `ImageTemplate`.

4.4 Remote Self-Hosting of Models

The computational demands of large pre-trained models can pose a challenge when using them for weak supervision development. To address this challenge, Alfred provides utilities for deploying and interacting with models on remote virtual instances or computing clusters. Additionally, Alfred implements a SSH-based tunneling service that ensures a secure local connection while preserving all Alfred functionality. The tunneling utility also simplifies deployment of the server on shared computing clusters, with the login node serving as a jump host for the computing node. This is particularly useful for using Alfred on centrally-managed shared computing clusters such as those at universities. Alfred’s built-in SSH tunneling is also capable of handling 2-factor authentication, which is common for shared clusters. To enable efficient communication between the client and server, Alfred uses gRPC, a high-performance, open-source remote procedure call framework. This enables Alfred to provide a seamless development experience for weak supervision development without the need for expensive local resources.

4.5 Mapping Responses to Votes

Another core piece of the Alfred system is the `Voter` class. Each `Voter` defines how to map model responses to votes for the true label of an example. The votes can be class labels or partial labels (e.g., attributes) specified by the users. The voting mechanism also relies on a matching function, which by default only casts a vote for an exact match. Users may provide their intended matching mechanisms such as uncased

matching or embedding similarity matching for more flexibility for each `Voter`. Furthermore, `Voter` can be contextually calibrated for the specific `Template` class to reduce model bias towards predicting certain answers. Recent studies show calibration can be helpful for many prompt-based tasks (Zhao et al., 2021; Smith et al., 2022). By calling ‘`Client.calibrate(Template, Voter)`,’ Alfred will calibrate the voting weights according to the strategy proposed by Zhao et al. and automatically apply the calibration during voting.

4.6 Label Models for Aggregating Votes

Alfred currently includes four label models for combining the votes from prompted labeling functions. The four label models are available to meet different use cases. The `MajorityVote` model is a baseline option suitable for fast development iteration, while the `NaiveBayes` model is recommended as the standard label model. Alfred also includes `NPLM` (Yu et al., 2022) (noisy partial label model) to support weak supervision from partial labels, which are labels that narrow down the possible set of classes but are not specific enough to vote on a single class. `FlyingSquid` (Fu et al., 2020) is the fourth model option and is recommended when `MajorityVote` is not accurate enough but more speed than `NaiveBayes` is needed. These label model classes have a unified interface, providing a consistent experience for users. After processing votes, the label model module generates probabilistic labels, represented as a distribution over the label space, for the given unlabeled dataset. Finally users can use the estimated probabilistic labels to train an end model for the downstream task.

5 Example Use Cases

In this section, we present two example use cases for how Alfred can be used to create training data for specific machine learning tasks using natural language prompts in both text and image domains. We measure the labeling quality by taking the top-1 accuracy of the estimated probabilistic labels given by the label model. The notebooks to reproduce these examples are in the Alfred repository.

5.1 Youtube Comment Spam Detection

Zero Shot	Prompted LFs	Prompted LFs+C
46.8	57.8	65.3

Table 1: Top-1 accuracy on YouTube spam detection. Zero Shot refers to prompting T0++ directly. +C means applying contextual calibration on the `Voter` objects.

In this experiment, we use Alfred to annotate the training split of YouTube spam detection dataset. (Alberto et al., 2015) We replicate the setup used by Smith et al. The prompts are translated from the code-based labeling functions provided by the WRENCH benchmark (Zhang et al., 2021), a comprehensive weak supervision benchmark. Alfred also includes a `WrenchBenchmarkDataset` abstraction for easily running this benchmark. In total, we define 10 prompted labeling functions with `StringTemplate` objects. Responses are mapped to votes using `Voter` objects. For this experiment, we use T0++ (Sanh et al., 2022) as the backbone model for Alfred. Following Smith et al., we also calibrate the responses from T0++ when voting using the contextual calibration strategy proposed by Zhao. Finally we aggregate the votes using the `NaiveBayes` label model to produce the probabilistic labels. Table 1 shows that Alfred makes reproducing the results of Smith et al. easy, demonstrating that the combination of weak supervision and calibration yield a dramatic improvement over zero-shot prompting alone.

5.2 Pet Breed Classification

Zero Shot	Prompted LFs
86.0	92.4

Table 2: Top-1 accuracy on Oxford-IIIT Pet breed classification. Zero Shot refers to prompting CLIP directly.

Traditionally, programmatic weak supervision for vision has been limited by the ability to express

supervision in code, relying on models such as object or attribute detectors to extract features and classify. However, these object detectors often depend heavily on supervised training data, becoming a bottleneck for applying programmatic weak supervision in various vision tasks. Fortunately, with large-pretrained vision-language model like CLIP (Radford et al., 2021), we are now able to express supervision with natural language. For this task, we develop prompts to classify 37 different breeds of pets from the Oxford-IIIT Pet dataset (Parkhi et al., 2012). We use CLIP-ViT/L-14 as the backbone model and developed three simple prompted LFs. The first two prompted LFs use templates “a photo of [[label]]” and “a photo of [[label]] [cat/dog]” where “[cat/dog]” is selected based on the breed. The third prompted LF produces a partial label with the template "a photo of [cat/dog]", encouraging fine-grained labels to match with the coarse-grained type detected by CLIP. We combine the votes using the `NPLM` label (Yu et al., 2022) to support weak supervision at various levels of granularity in the label space. Table 2 shows that this multi-granular weak supervision provides a nice boost over zero-shot prompting alone.

6 Discussion and Future Work

This paper introduces Alfred, a prototype for the next generation of programmatic weak supervision systems that leverage the potential of large pre-trained models. Alfred complements the existing ecosystem of large-pretrained-model toolkits, offering optimized inference throughput, a smooth local development experience, and compatibility with vision-language models to support image annotation tasks. Alfred represents a notable advancement in the domain of programmatic weak supervision, as it enables users to express their domain-specific knowledge and heuristics with flexible natural language prompts for language and vision-language models. This approach can be more user-friendly than conventional PWS systems, which requires expert programming of weak supervision sources. Our objective is for Alfred to serve as the infrastructure and experimentation platform for many future weak supervision research projects and applications. Furthermore, we plan to extend Alfred’s capabilities to accommodate a wider range of multimodal large pre-trained models, such as Whisper (Radford et al., 2022) and LayoutLMs (Xu et al., 2020b,a; Huang et al., 2022).

Limitations

Alfred is a prototype for the second generation of PWS systems, which incorporate large pre-trained models. However, there are some potential limitations to consider. As with all PWS approaches, application quality is limited by the quality of the weak supervision sources used to vote on the labels. In this case of prompted labeling functions, this depends on how well suited the prompts and model are to the task and domain. If they are not well suited, then additional fine-tuning of the prompted models will be necessary. Compared with traditional labeling functions written in code, understanding when and why labeling functions fail on certain examples can be particularly challenging. Methods for explanations such as minimal contrastive edits (Ross et al., 2021) can potentially help address this limitation. We plan to explore incorporating such methods into Alfred.

Ethics Statement

One major concern for Alfred is the potential for biased or unfair labeling. Large pre-trained models are trained on massive datasets, which can reflect societal biases and inequalities. Consequently, supervision generated by these models can perpetuate and amplify these biases, leading to discrimination or unfair treatment in downstream applications. Therefore, it is essential to carefully consider the quality and representativeness of the backbone model for Alfred, as well as the prompts used for labeling data. To address potential labeling biases, human oversight and auditing are needed during the development loop to spot and correct any issues. While Alfred has the potential to enhance the efficiency of programmatic data labeling, it is crucial to carefully consider and address potential ethical challenges.

Acknowledgments

We appreciate the helpful comments and discussion with Andrew Yuan, Avi Trost, Nihal Nayak and Zheng-Xin Yong. This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under agreement number FA8750-19-2-1006. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and

should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) or the U.S. Government. We gratefully acknowledge support from Google and Cisco. Disclosure: Stephen Bach is an advisor to Snorkel AI, a company that provides software and services for weakly supervised machine learning.

References

- Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. 2015. Tubespm: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 138–143. IEEE.
- Simran Arora, Avaniika Narayan, Mayee F Chen, Laurel J Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. Ask me anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441*.
- Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. [Promptsources: An integrated development environment and repository for natural language prompts](#).
- 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.
- Harrison Chase. 2022. [LangChain](#).
- Mayee F Chen, Daniel Y Fu, Dyah Adila, Michael Zhang, Frederic Sala, Kayvon Fatahalian, and Christopher Ré. 2022. Shoring up the foundations: Fusing model embeddings and weak supervision. In *Uncertainty in Artificial Intelligence*, pages 357–367. PMLR.

- Yew Ken Chia, Lidong Bing, Soujanya Poria, and Luo Si. 2022. Relationprompt: Leveraging prompts to generate synthetic data for zero-shot relation triplet extraction. *arXiv preprint arXiv:2203.09101*.
- Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. [OpenPrompt: An open-source framework for prompt-learning](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland. Association for Computational Linguistics.
- Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*.
- Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A community library for natural language processing. *arXiv preprint arXiv:2109.02846*.
- Laurel Orr. 2022. Manifest. <https://github.com/HazyResearch/manifest>.
- Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.
- 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, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- 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 VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.
- Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.
- Alexis Ross, Ana Marasović, and Matthew Peters. 2021. [Explaining NLP models via minimal contrastive editing \(MiCE\)](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852, Online. Association for Computational Linguistics.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. [Multi-task prompted training enables zero-shot task generalization](#). In *International Conference on Learning Representations*.
- Timo Schick and Hinrich Schütze. 2021a. [Few-shot text generation with natural language instructions](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Timo Schick and Hinrich Schütze. 2021b. [Generating datasets with pretrained language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ryan Smith, Jason A Fries, Braden Hancock, and Stephen H Bach. 2022. Language models in the loop: Incorporating prompting into weak supervision. *arXiv preprint arXiv:2205.02318*.
- Thomas Wolf Philipp Schmid Zachary Mueller Sourab Mangrulkar Sylvain Gugger, Lysandre Debut. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. <https://github.com/huggingface/accelerate>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- 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.
- Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. **Generating data to mitigate spurious correlations in natural language inference datasets**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2660–2676, Dublin, Ireland. Association for Computational Linguistics.
- Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.
- Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020b. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.
- Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*.
- Peilin Yu, Tiffany Ding, and Stephen H. Bach. 2022. Learning from multiple noisy partial labelers. In *Artificial Intelligence and Statistics (AISTATS)*.
- Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*.
- Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022a. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*.
- Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. **WRENCH: A comprehensive benchmark for weak supervision**. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022b. Prboost: Prompt-based rule discovery and boosting for interactive weakly-supervised learning. *arXiv preprint arXiv:2203.09735*.
- Fang Zhao. 2022. **Auto-correction dans un analyseur neuronal par transitions : un comportement factice ? (self-correction in a transition-based neural parser : a spurious behaviour ?)**. In *Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 2 : 24e Rencontres Etudiants Chercheurs en Informatique pour le TAL (RECITAL)*, pages 20–32, Avignon, France. ATALA.
- Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

OpenICL: An Open-Source Framework for In-context Learning

Zhenyu Wu^{♦†*}, Yaoxiang Wang^{♣†*}, Jiacheng Ye^{♠†}
Jiangtao Feng[◇], Jingjing Xu[◇], Yu Qiao[◇], Zhiyong Wu^{◇‡}
◇Shanghai AI Laboratory ♦ East China Normal University
♣Xiamen University ♠The University of Hong Kong

carsonye@cs.hku.hk, {wuzhenyu, wangyaoxiang}@pjlab.org.cn
{fengjiangtao, xujingjing, qiaoyu, wuzhiyong}@pjlab.org.cn

Abstract

In recent years, In-context Learning (ICL) has gained increasing attention and emerged as the new paradigm for large language model (LLM) evaluation. Unlike traditional fine-tuning methods, ICL instead adapts the pre-trained models to unseen tasks *without* any parameter updates. However, the implementation of ICL is sophisticated due to the diverse retrieval and inference methods involved, as well as the varying pre-processing requirements for different models, datasets, and tasks. A unified and flexible framework for ICL is urgently needed to ease the implementation of the aforementioned components. To facilitate ICL research, we introduce OpenICL, an open-source toolkit for ICL and LLM evaluation. OpenICL is research-friendly with a highly flexible architecture that users can easily combine different components to suit their needs. It also provides various state-of-the-art retrieval and inference methods to streamline the process of adapting ICL to cutting-edge research. The effectiveness of OpenICL has been validated on a wide range of NLP tasks, including classification, QA, machine translation, and semantic parsing. As a side-product, we found OpenICL to be an efficient yet robust tool for LLMs evaluation. OpenICL is released at <https://github.com/Shark-NLP/OpenICL>.

1 Introduction

The rise of large language models (LLMs) (Brown et al., 2020; Zhang et al., 2022a; Scao et al., 2022) has shown impressive emergent In-Context Learning (ICL) ability (Wei et al., 2022a). Different from finetuning which requires parameter updates, ICL can perform inference with model parameters frozen. ICL sidesteps the resource-intensive nature of fine-tuning, yet still yields comparable results

to fine-tuned models in specific tasks (Zhao et al., 2021; Lu et al., 2022; Gao et al., 2021a). However, we observed a lack of a unified framework for ICL. Implementations from existing projects are often high-customized to their own needs, thus making further development and comparisons with previous approaches a challenge.

The basic ICL pipeline contains two steps: retrieval and inference. Given a testing input X' , in the retrieval stage, several examples from the training set are retrieved as in-context demonstrations. In the inference stage, these demonstrations are prepended before X' and fed into the LLM to generate the prediction. Researchers have explored various methods for both retrieval (e.g., BM25 (Robertson and Zaragoza, 2009), TopK (Liu et al., 2022; Gao et al., 2021a) and VoteK (Su et al., 2022)) and inference (e.g., perplexity-based (Brown et al., 2020), channel-based (Min et al., 2022), and Chain-of-thoughts (Wei et al., 2022b)). However, these methods are often implemented under different frameworks, and/or evaluated using different LLMs and tasks. These inconsistencies make systematic evaluations and comparisons of various methods challenging, thus hindering the development of better ICL methods.

To address this issue, we present OpenICL, an open-source and easy-to-use toolkit for ICL. OpenICL has many state-of-the-art retrieval and inference methods built in to facilitate systematic comparison and fast research prototyping. OpenICL also provides a unified and flexible interface for the development and evaluation of new ICL methods. Users can easily incorporate different retrieval and inference methods, as well as different prompt instructions, into their pipelines. To validate OpenICL’s implementation and design, we use OpenICL to evaluate LLMs on several NLP tasks, including classification, question answering, translation, and semantic parsing. Our contributions are summarized as follows:

*Work done while interning at Shanghai AI Lab.

†Equal Contribution.

‡Corresponding Author.

- We propose OpenICL, an easy-to-use and extensible ICL framework for zero-/few-shot evaluation of language models
- OpenICL provides a wide range of ICL methods, LLMs, and tasks, requiring as little as a few lines of code to use and paving the way for more extensions in the future.
- We provide complete tutorials to walk users through the framework, thus facilitating research and development of ICL.

2 Related Work

In-context Learning Besides the classic “pre-train and fine-tune” paradigm, [Brown et al. \(2020\)](#) proposed In-context learning (ICL), a new paradigm that leverages pre-trained language models to perform new tasks without any gradient-based training. It appends a small number of training examples as prompts before the test input, and have shown to be able to improve LLMs’ performance in few-shot scenarios and generalize to a wide range of downstream tasks, such as information retrieval ([Tay et al., 2022](#)), fact checking ([Rae et al., 2021](#)), commonsense reasoning ([Geva et al., 2021](#)), arithmetic reasoning ([Cobbe et al., 2021](#)), machine translation ([Agrawal et al., 2022](#); [Lin et al., 2021a](#)), and data generation ([Ye et al., 2022](#)), etc.

Aside from those early successes, researchers have developed more sophisticated ICL methods that require some intermediate reasoning steps. Among them, chain-of-thoughts (CoT) is the first attempt that significantly surpasses the previous state-of-the-art methods on many reasoning tasks ([Wei et al., 2022b](#)). After that, different variants of CoT have been proposed to strengthen its performance, such as Self-Ask ([Press et al., 2022](#)), iCAP ([Wang et al., 2022](#)), Least-to-Most prompting ([Zhou et al., 2022](#)), and Selection-Inference ([Zhang et al., 2022b](#); [Fu et al., 2022](#)).

Despite the surprising performance, ICL has been criticized for being very sensitive to the choice and ordering of in-context examples ([Zhao et al., 2021](#); [Lu et al., 2022](#)). To address this problem, different criterion and context construction methods have been proposed. [Gao et al. \(2021a\)](#) and [Liu et al. \(2022\)](#) select examples that are closer to the test input in the embedding space; a line of work ([Su et al., 2022](#); [Levy et al., 2022](#); [Ye et al., 2023](#)) select the most representative examples in

the training set to encourage diversity of in-context examples; [Wu et al. \(2022\)](#) observe that Minimum Description Length (MDL) principle can be an effective criterion for in-context example selection.

Prompt Learning Prompt learning ([Liu et al., 2021](#)) is a special case of ICL without any in-context examples. Prompt learning comprises various topics including manual template engineering ([Petroni et al., 2019](#); [Brown et al., 2020](#)), automated template learning ([Wallace et al., 2019](#); [Shin et al., 2020](#); [Li and Liang, 2021](#)), and answer engineering ([Gao et al., 2021b](#); [Schick and Schütze, 2021](#)). We refer the readers to the usage of OpenPrompt ([Ding et al., 2021](#)) which is a toolkit specially designed for prompt learning. In comparison, OpenICL focuses more on integrating various exemplar retrieving approaches and inference strategies for in-context learning. Note that OpenICL can also seamlessly support prompt learning by setting the number of in-context examples to zero and specifying the manual or pre-searched prompt templates by OpenPrompt for different tasks.

3 OpenICL

In this section, we first explain OpenICL’s design principles. Then, we will briefly describe OpenICL’s two major components, namely, the `Retriever` and `Inferencer`.

3.1 Design Principles

The design principle of OpenICL is to facilitate in-context learning research and enable efficient and robust large language model evaluation. In detail, we consider the following principles:

[P1: Modularity] Since ICL is a fast-evolving research field, the design of OpenICL should be decoupled such that different components can be easily modified to support latest methods and/or combined to suit various tasks and application needs.

[P2: Efficiency] Nowadays, large language models can have hundreds of billions of parameters. To support inference at such a massive scale, OpenICL should be optimized to enable efficient parallel inference.

[P3: Generality] ICL has been widely used in all fields in NLP, so OpenICL needs a flexible interface that enables it to work with various LLMs, tasks, retrieval methods, and inference approaches.

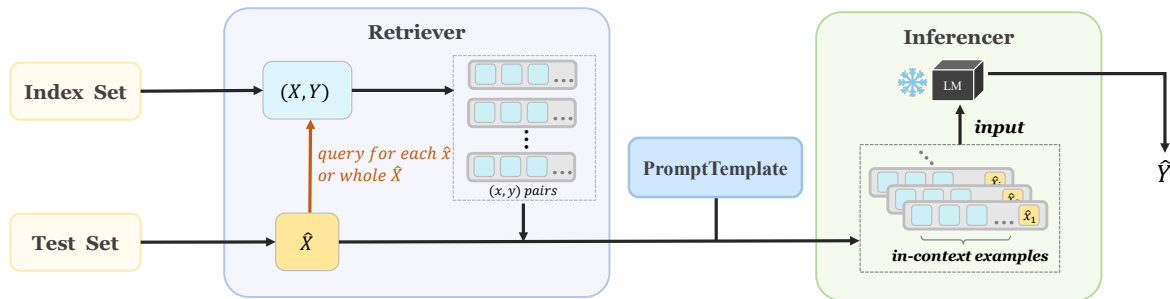


Figure 1: Overview of the architecture in OpenICL. OpenICL first obtains proper in-context examples from an index set for each test input or for the whole test set via retrieval methods (e.g., TopK or VoteK) specified by the users. Then the in-context examples and test input are concatenated into a single sequence based on the provided prompt template. Finally, all the prompts are fed into the language model to infer the output through defined inference strategies (e.g., Chain-of-thought).

3.2 Architecture Overview

Figure 1 overviews OpenICL’s architecture. For each input \hat{x} from the test set \hat{X} , the `Retriever` retrieves several (x, y) pairs (represented as one row in the dashed box) from an index set (X, Y) as \hat{x} ’s in-context examples. These examples, as well as \hat{x} , are then formatted according to the user-defined prompt template and concatenated to form a text sequence. After that, the `Inferencer` digests these sequences and fed them into the LLMs to obtain the model prediction \hat{Y} .

3.3 Modularity

To satisfy Principle P1, OpenICL adopts a loosely-coupled design between components. These components separate the data pre-processing, retrieval, and inference processes with very flexible interfaces that allow easy customization to fit specific needs. Two major components are detailed below:

Retriever `Retriever` is responsible for retrieving in-context examples from the pre-existing training data. This module supports both corpus-level (i.e., only retrieving one group of examples for the whole test set) and instance-level (i.e., retrieving examples for each testing input individually) retrieval methods. OpenICL primarily supports learning-free retrieval methods as follows:

- **Random:** Early practice (Brown et al., 2020) of ICL often randomly select examples to construct the context. Although Random brings high variance for ICL performance, it is still the popular choice when there are only a few demonstrations available (Wei et al., 2022b; Zhao et al., 2021).

- **Heuristic method:** To overcome the disadvantage of Random, various semantic similarity based retrieval methods have been proposed and shown great promise, such as BM25 (Robertson and Zaragoza, 2009), TopK (Liu et al., 2022; Gao et al., 2021a), and VoteK (Su et al., 2022).
- **Model-based method:** More recently, researchers have explored using models’ confidence in the output to select and order examples, such as entropy (Lu et al., 2022) and MDL (Wu et al., 2022).

OpenICL has implemented the existing methods above to facilitate future research and systematic comparison. Furthermore, the flexibility of the `Retriever` module allows practitioners to select the retrieval method and make further modification that best suits their task and data. The interface of `Retriever` also allows users to pack those in-context examples and use them somewhere else.

Inferencer `Inferencer` invokes the pre-trained language model to generate predictions based on the concatenation of in-context examples and testing input. The `Inferencer` supports various inference methods:

- **Direct:** Brown et al. (2020) use tokens in the vocabulary to represent candidate answers and select the final prediction using the one with the highest probability.
- **Perplexity:** (Brown et al., 2020) compute the sentence perplexity of the sequence concatenation of input and candidate answers and

select the final prediction using the one with the lowest perplexity.

- Channel: Min et al. (2022) proposed to utilize channel models (Yu et al., 2016; Yee et al., 2019) to compute the conditional probability in a reversed direction, i.e., estimating the likelihood of input query given the label.

The flexibility of `Inferencer` also allows users to recursively invoke it to support multi-stage ICL methods, such as chain-of-thought (Wei et al., 2022b) and selection-inference (Creswell et al., 2022). Additionally, `Inferencer` can be augmented with a scorer to evaluate its prediction.

3.4 Efficiency

To satisfy Principle P2, we equip OpenICL with various parallelism techniques to enable efficient inference for large-scale models.

Data Parallel Data parallel (Li et al., 2020) is a common technique used in parallel computing to improve the efficiency of large-scale computation tasks. OpenICL implements data parallelism to improve the performance of both the retrieval and inference steps. During retrieval and inference, data is divided into smaller batches for processing. Additionally, for models that can fit into GPU’s VRAM, OpenICL implements data parallelism by sharding the data across multiple GPUs and performing parallel inference on each GPU with a complete copy of the model. This significantly increases the inference speed when working with large datasets.

Model Parallel In the era of LLMs, models often have billions or hundreds of billions of parameters that exceed modern GPUs’ capacity. To handle this problem, we resort to model parallel (Shoeybi et al., 2019): a parallel computing technique that divides a large deep learning model into smaller sub-models, each of which can be run on a separate GPU. OpenICL supports model parallelism that users can easily parallelize their models with minimal modification to the code. Currently, we support Megatron (Shoeybi et al., 2019) and Zero (Rajbhandari et al., 2019).

3.5 Generality

To satisfy Principle P3, OpenICL is designed to maximize users’ productivity by supporting a wide range of models, tasks, and methods:

[Model] OpenICL supports both decoder-only LMs (e.g., GPT family (Radford and Narasimhan, 2018; Radford et al., 2019; Black et al., 2021; Wang and Komatsuzaki, 2021; Black et al., 2022), and encoder-decoder-based LMs (e.g., T5 (Raffel et al., 2020))). We also provide two alternatives for accessing the model: users can directly load model checkpoints for evaluation or access a model via API (e.g., OpenAI’s GPT-3 series models; Brown et al. 2020; Chen et al. 2021; Ouyang et al.).¹

[Tasks] With the help of OpenICL, users can easily conduct experiments on both classification and generation tasks. OpenICL integrates HuggingFace’s `datasets`² such that users can access and download thousands of NLP tasks with ease.

[Methods] As aforementioned, OpenICL provides broad support for ICL methods that cover both retrieval and inference. Furthermore, OpenICL offers the flexibility to return the results of the `Retriever` and `Inferencer` in a step-by-step manner, making it easy to integrate these intermediate results into other projects.

4 Toolkit Walkthrough

In this section, we demonstrate OpenICL by walking readers through several typical ICL use cases.

Example 1. We first demonstrate how to use OpenICL to develop a typical ICL pipeline for language classification using a few lines of code and conduct evaluation on the popular sentiment classification dataset SST-2 (Socher et al., 2013). As shown in Figure 2, the pipeline begins with a `DatasetReader` which loads the dataset given its name on HuggingFace Dataset Hub³ or local file path. Users need to specify the names of columns where the input (“*text*”) and output (“*label*”) are stored. Secondly, a customized `PromptTemplate` is instantiated with a dictionary that defines the prompts for each class label. The placeholder `</E>` and `</Q>` will be replaced by in-context examples and testing input, separately. After that, we initiate the retriever based on TopK (Liu et al., 2022) and set the number of in-context examples to 8 (“*ice_num = 8*”). We select perplexity-based method to initiate the inferencer and use GPT2-XL as the LLM. Having

¹<https://openai.com/api/>

²<https://github.com/huggingface/datasets>

³<https://huggingface.co/datasets>

```

1 from openicl import DatasetReader, PromptTemplate
2 from openicl import TopkRetriever, PPLInferencer, AccEvaluator
3
4 # Load dataset
5 data = DatasetReader('gpt3mix/sst2', input_columns=['text'], output_column='label')
6
7 # Define the prompt template for the task
8 tp_dict = { 0: '</E> Positive Movie Review: </Q>',
9             1: '</E> Negative Movie Review: </Q>' }
10 template = PromptTemplate(tp_dict, {'text': '</Q>'}, ice_token='</E>')
11
12 # Initiate the retriever and inferencer
13 retriever = TopkRetriever(data, ice_num=8)
14 inferencer = PPLInferencer(model_name='gpt2-xl')
15
16 # Run inference and calculate score
17 predictions = inferencer.inference(retriever, ice_template=template)
18 score = AccEvaluator().score(predictions=predictions, references=data.references)

```

Figure 2: Illustration of Example 1 which evaluates the ICL performance of GPT2-XL (1.5B) on SST-2 dataset with PPL inference strategy.

```

1 from datasets import load_dataset
2 from openicl import DatasetReader, PromptTemplate
3 from openicl import RandomRetriever, GenInferencer, BleuEvaluator
4
5 dataset = load_dataset("wmt16", 'de-en').map(lambda example: example['translation'])
6
7 data = DatasetReader(dataset, input_columns=['de'], output_column='en')
8
9 template = PromptTemplate('</E> German:</German> \n English: </English>',
10                          {'de': '</German>', 'en': '</English>'}, ice_token='</E>')
11
12 retriever = RandomRetriever(data, ice_num=8)
13
14 # Inference by direct generation
15 inferencer = GenInferencer(model_name='facebook/xglm-7.5B')
16 predictions = inferencer.inference(retriever, ice_template=template)
17
18 # calculate Bleu
19 score = BleuEvaluator().score(predictions=predictions, references=data.references)

```

Figure 3: Illustration of Example 2 that evaluates the ICL performance of XGLM (7.5B) on WMT16 (de-en) dataset with direct inference strategy.

```

1 from openicl import DatasetReader, PromptTemplate, BM25Retriever, CoTInferencer
2
3 data = DatasetReader('gsm8k', name='main',
4                     input_columns=['question'], output_column='answer')
5
6 template = PromptTemplate('</E> Question: </Q> \n Answer: </A>',
7                          {'question': '</Q>', 'answer': '</A>'},
8                          ice_token='</E>')
9
10 retriever = BM25Retriever(data, ice_num=4)
11
12 # Inference by Chain-of-Thought
13 cot_list=["Let's think step by step.",
14          "\nTherefore, the answer (arabic numerals) is"]
15
16 inferencer = CoTInferencer(cot_list=cot_list, api_name='gpt3')
17 predictions = inferencer.inference(retriever, ice_template=template)

```

Figure 4: Illustration of Example 3, which evaluates the ICL performance of *text-davinci-003* version of GPT-3 (175B) on GSM8K dataset with Chain-of-thought inference strategy.

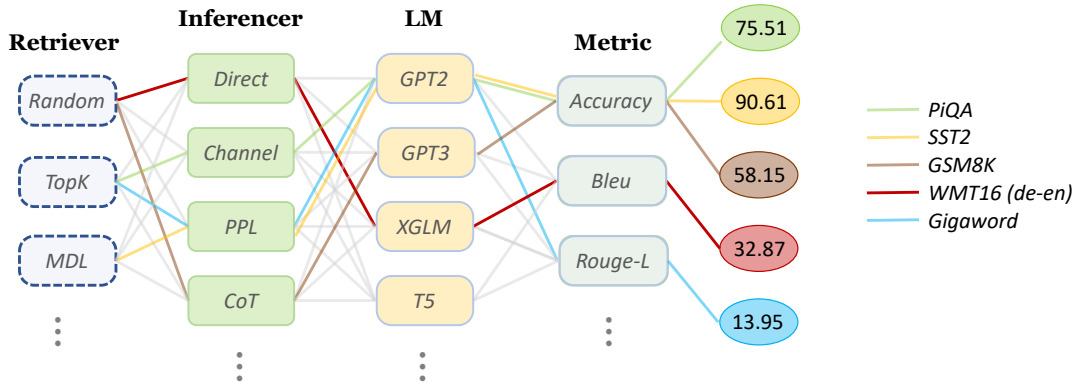


Figure 5: Evaluation results. We conduct experiments on five representative tasks with OpenICL and use different retrievers, inferencers, language models, and other components. In terms of model usage, we adopt GPT-Neo (2.7B) for SST2, PiQA, and Gigaword, XGLM (7.5B) for WMT16 (de-en), and *text-davinci-003* version of GPT-3 (175B) for GSM8K.

all these been set, we can run the inference by invoking the inferencer (line 17) and calculating the accuracy of the model’s prediction(line 18).

Example 2. Figure 3 shows how to use OpenICL to work with generation problems. We consider the popular machine translation dataset WMT16 (Borjar et al., 2016). As in Example 1, we can easily load the dataset, define the prompt template, and initiate the retriever, by feeding new parameters to the function, respectively. The major API difference from Example 1 is that (i) we add some pre-processing for the translation task (line 5); (ii) *PPLInferencer* is replaced by inferencer tailored for generation (line 16); (iii) we use BLEU to evaluate model performance.

Example 3. OpenICL also supports more advanced ICL methods, as shown in Figure 4. Users can seamlessly switch to CoT by only modifying two lines of code: line 14 defines the template for CoT and line 15 initiates the inferencer with GPT3 using OpenAI’s API. Similar multi-step ICL methods such as Self-Consistency (Wang et al., 2022) and Selection-Inference (Creswell et al., 2022) can also be easily implemented by inheriting the superclass *Inferencer* designed in OpenICL.

5 Evaluation

To demonstrate OpenICL’s flexibility we conducted experiments on a diverse set of datasets, LLMs, and ICL methods. We consider PiQA (Bisk et al., 2019) for commonsense reasoning, SST-2 (Socher et al., 2013) for sentiment analysis, GSM8K (Cobbe et al.,

2021) for arithmetic reasoning, WMT16 de-en (Borjar et al., 2016) for machine translation and Gigaword (Napoles et al., 2012) for summarization. We’ve also tested various LLMs, including GPT-Neo (2.7B) (Black et al., 2021; Gao et al., 2020), *text-davinci-003* version of GPT-3 (175B), and XGLM (7.5B) (Lin et al., 2021b). We use OpenAI’s official API⁴ to access GPT-3. The detailed setups and results are shown in Figure 5. As we can see, components of OpenICL can be easily chained to support different evaluation needs and replicate results of state-of-the-art methods.

6 Conclusion

We present OpenICL, an open-source toolkit for In-context learning. OpenICL provides a convenient and flexible interface for in-context learning practice and research. Our modular design allows it to support a wide range of LLMs, tasks, and ICL methods with ease. We implement both model parallelism and data parallelism to make inference of large models more efficient. OpenICL is highly extensible, and we will continue to update it to keep pace with the latest research. Despite the promising results, ICL is still in its early stages, and many challenges remain. We believe OpenICL will be a valuable resource for researchers and practitioners to facilitate their research and development.

⁴<https://openai.com/api/>

References

- Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2022. [In-context examples selection for machine translation](#).
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. [PIQA: Reasoning about Physical Commonsense in Natural Language](#). *arXiv e-prints*, page arXiv:1911.11641.
- Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. [Gpt-neox-20b: An open-source autoregressive language model](#).
- Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. [GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow](#). If you use this software, please cite it using these metadata.
- Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Liane Guillou, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Aurélie Névéol, Mariana Neves, Pavel Pecina, Martin Popel, Philipp Koehn, Christof Monz, Matteo Negri, Matt Post, Lucia Specia, Karin Verspoor, Jörg Tiedemann, and Marco Turchi, editors. 2016. [Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers](#). Association for Computational Linguistics, Berlin, Germany.
- 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](#). *CoRR*, abs/2005.14165.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. [Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning](#). *arXiv e-prints*, page arXiv:2205.09712.
- Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. [OpenPrompt: An Open-source Framework for Prompt-learning](#). *arXiv e-prints*, page arXiv:2111.01998.
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. [Complexity-based prompting for multi-step reasoning](#). *CoRR*, abs/2210.00720.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. [The pile: An 800gb dataset of diverse text for language modeling](#). *arXiv preprint arXiv:2101.00027*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. [Making pre-trained language models better few-shot learners](#). 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 3816–3830.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021b. [Making pre-trained language models better few-shot learners](#). 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 3816–3830, Online. Association for Computational Linguistics.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies](#). *arXiv e-prints*, page arXiv:2101.02235.
- Itay Levy, Ben Bogin, and Jonathan Berant. 2022. [Diverse demonstrations improve in-context compositional generalization](#). *arXiv preprint arXiv:2212.06800*.
- Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. [Pytorch distributed: Experiences on accelerating data parallel training](#).
- 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.
- Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O’Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin

- Stoyanov, and Xian Li. 2021a. [Few-shot learning with multilingual language models](#). *CoRR*, abs/2112.10668.
- Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O’Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin Stoyanov, and Xian Li. 2021b. [Few-shot learning with multilingual language models](#). *CoRR*, abs/2112.10668.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.
- 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](#). *CoRR*, abs/2107.13586.
- Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.
- Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. [Noisy channel language model prompting for few-shot text classification](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.
- Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. [Annotated Gigaword](#). In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada. Association for Computational Linguistics.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. [Language models as knowledge bases?](#) In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. [Measuring and narrowing the compositionality gap in language models](#). *CoRR*, abs/2210.03350.
- Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsim-poukelli, Nikolai Grigorev, Doug Fritz, Thibault Sotiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d’Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorraine Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. [Scaling Language Models: Methods, Analysis & Insights from Training Gopher](#). *arXiv e-prints*, page arXiv:2112.11446.
- 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.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. [Zero: Memory optimization towards training A trillion parameter models](#). *CoRR*, abs/1910.02054.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman

- Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100.
- Timo Schick and Hinrich Schütze. 2021. [It’s not just size that matters: Small language models are also few-shot learners](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. [Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism](#). *arXiv e-prints*, page arXiv:1909.08053.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. [Megatron-lm: Training multi-billion parameter language models using model parallelism](#). *CoRR*, abs/1909.08053.
- 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.
- Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*.
- Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. [Transformer Memory as a Differentiable Search Index](#). *arXiv e-prints*, page arXiv:2202.06991.
- Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. [Universal adversarial triggers for attacking and analyzing NLP](#). 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 2153–2162, Hong Kong, China. Association for Computational Linguistics.
- Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>.
- Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *The 2022 Conference on Empirical Methods for Natural Language Processing*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. [Self-Consistency Improves Chain of Thought Reasoning in Language Models](#). *arXiv e-prints*, page arXiv:2203.11171.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022b. [Chain of thought prompting elicits reasoning in large language models](#). *CoRR*, abs/2201.11903.
- Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2022. [Self-adaptive in-context learning](#).
- Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2022. [ProGen: Progressive zero-shot dataset generation via in-context feedback](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3671–3683, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. [Compositional exemplars for in-context learning](#). *arXiv preprint arXiv:2302.05698*.
- Kyra Yee, Nathan Ng, Yann N. Dauphin, and Michael Auli. 2019. [Simple and effective noisy channel modeling for neural machine translation](#). *CoRR*, abs/1908.05731.
- Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomáš Kociský. 2016. [The neural noisy channel](#). *CoRR*, abs/1611.02554.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022a. [Opt: Open pre-trained transformer language models](#).
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models. *CoRR*, abs/2210.03493.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. [Calibrate Before Use: Improving Few-Shot Performance of Language Models](#). *arXiv e-prints*, page arXiv:2102.09690.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. [Least-to-most prompting enables complex reasoning in large language models](#). *ArXiv preprint*, abs/2205.10625.

Self-Supervised Sentence Polishing by Adding Engaging Modifiers

Zhexin Zhang¹, Jian Guan¹, Xin Cui², Yu Ran², Bo Liu² and Minlie Huang^{1*}

¹The CoAI group, DCST; ¹Institute for Artificial Intelligence; ¹State Key Lab of Intelligent Technology and Systems;

¹Beijing National Research Center for Information Science and Technology; ¹Tsinghua University, Beijing 100084, China.

²Platform and Content Group, Tencent Technology Co., Ltd.

{zx-zhang22,j-guan19}@mails.tsinghua.edu.cn, {kimicui,ryanran,ustbliu}@tencent.com

aihuang@tsinghua.edu.cn

Abstract

Teachers often guide students to improve their essays by adding engaging modifiers to polish the sentences. In this work, we present the first study on automatic sentence polishing by adding modifiers. Since there is no available dataset for the new task, we first automatically construct a large number of parallel data by removing modifiers in the engaging sentences collected from public resources. Then we fine-tune LongLM (Guan et al., 2022) to reconstruct the original sentences from the corrupted ones. Considering that much overlap between inputs and outputs may bias the model to completely copy the inputs, we split each source sentence into sub-sentences and only require the model to generate the modified sub-sentences. Furthermore, we design a retrieval augmentation algorithm to prompt the model to add suitable modifiers. Automatic and manual evaluation on the auto-constructed test set and real human texts show that our model can generate more engaging sentences with suitable modifiers than strong baselines while keeping fluency. We deploy the model at <http://coai.cs.tsinghua.edu.cn/static/polishSent/>. A demo video is available at <https://youtu.be/Y6gFH0gSv8Y>.

1 Introduction

Teachers’ guidance is necessary for students to improve their essays in primary and secondary writing education. For example, teachers can point out potential logical errors and incoherence issues, and polish sentences to improve the engagingness of the essays. A typical way to polish sentences is to add engaging modifiers (e.g., from “*I ate a pear*” to “*I ate a big pear enjoyably*”), which usually are adjectives or adverbs that enhance the meaning of a sentence (Witte and Faigley, 1981). Since an essay usually contains tens of sentences, it is a heavy burden for teachers to polish each one. To reduce

*Corresponding author

Original Sentences	Corrupted Sentences
先是轻盈的雨滴轻轻地滴落，发出悦耳的“叮咚”声，就像是乐曲的前奏。(First, light raindrops drip softly, emitting a pleasant “ding-dong” sound, like a prelude to the music.)	先是雨滴轻轻地滴落，发出“叮咚”声，就像是乐曲的前奏。(First, raindrops drip softly, emitting a “ding-dong” sound, like a prelude to the music.)
情不自禁地哼起那首《乡间的小路》，抛开了一切繁重的心事，直到夜幕降临。(I can’t help but hum the song <i>Country Road</i> , leaving aside all heavy thoughts until nightfall.)	哼起那首《乡间的小路》，抛开了一切繁重的心事，直到夜幕降临。(I hum the song <i>Country Road</i> , leaving aside all heavy thoughts until nightfall.)

Table 1: Examples of automatically constructed data. After collecting original sentences, we corrupt them to construct less engaging sentences by removing the modifiers that are in a vocabulary of engaging words (marked in red).

teachers’ workload and enable students to improve their essays independently, we present a new study on *automatic sentence polishing*, which requires polishing a given sentence given its context. The goal of polishing a sentence is to make the sentence more expressive, attractive and engaging. We only consider inserting modifiers for polishing in this work, and leave other types of polishing to future work (e.g., replacing words or rephrasing the sentence). The challenges of the new task mainly manifest in the following two folds: (1) finding the words that can be modified; and (2) deciding suitable modifiers for those words.

Considering that there are no available parallel data for this new task, we propose a self-supervised learning approach using automatically constructed training data. We firstly collect a large number of engaging sentences from public books and student essays in Chinese, and then corrupt the sentences to construct less engaging ones by removing the modifiers in them, as exemplified in Table 1. We

learn a generation model for sentence polishing by training it to reconstruct the original sentences from the corrupted ones. To alleviate the model’s tendency to completely copy inputs as generation outputs, we train the model to generate only the changed sub-sentences split by commas (e.g., “I can’t help but hum the song Country Road” in the second example). Furthermore, we propose a novel retrieval augmentation algorithm to improve the correctness of added modifiers by retrieving suitable pairs of modifiers and modified words from the training set as additional inputs.

Automatic and manual evaluation on the auto-constructed test set and real human texts show that our model can generate more engaging sentences with suitable modifiers and comparable fluency than strong baselines. Furthermore, we build a website to enable real-time interaction with our deployed model, where a user can upload a Chinese sentence with its context and get the retrieval result along with the polished sentence.

2 Related Work

2.1 Constrained Text Generation

Automatic sentence polishing can be regarded as a kind of constrained text generation task (Garbacea and Mei, 2022), which requires generating coherent text that meets given constraints. Typical constrained generation tasks span from machine translation (Yang et al., 2020), summarization (Paulus et al., 2018), sentence generation from input concepts (Lin et al., 2020a), story generation from input phrases (Rashkin et al., 2020) or events (Ammanabrolu et al., 2020). Previous studies usually adopt the encoder-decoder framework (Sutskever et al., 2014) equipped with the attention mechanism (Bahdanau et al., 2015) to deal with constrained generation tasks. Recently, large-scale pretraining models based on the Transformer model (Vaswani et al., 2017) such as BART (Lewis et al., 2020) and LongLM (Guan et al., 2022) achieve more surprising performance (Lin et al., 2020b) although they are still far from humans (Lin et al., 2020a).

2.2 Text-Editing Models

There is significant overlap between inputs and outputs in many constrained generation tasks such as grammatical error correction (Omelianchuk et al., 2020) and sentence polishing in this work. When applying the vanilla encoder-decoder framework

	Train	Val	Test _{Auto}	Test _{Real}
# Examples	143,185	17,898	1000	1000
Avg. M Len	29.33	29.41	29.48	28.71
Avg. S Len	37.89	37.76	38.31	41.92
Avg. N Len	29.22	29.38	27.89	27.37
Avg. T Len	42.40	42.22	42.79	N/A

Table 2: Statistics of the dataset. *Len* is the abbreviation of *Length*. **Train**, **Val** and **Test_{Auto}** are the auto-constructed training, validation and test sets, respectively. **Test_{Real}** is the test set from real human-written sentences. We compute the length by counting the number of Chinese characters.

to such tasks, the models tend to directly copy the input without modification and it seems wasteful to generate the whole output text from scratch (Malmi et al., 2019). Text-editing models are proposed to address this issue, which usually conduct token-wise prediction for how to edit the token. LaserTagger (Malmi et al., 2019) presented three editing types including retaining the token, deleting the token, and inserting tokens before the token using a fixed phrase vocabulary obtained from the training set. Felix (Mallinson et al., 2020) further adopted a pointer network to learn to reorder the input tokens, and utilized a pretrained masked language model to predict inserted tokens. Seq2Edits (Stahlberg and Kumar, 2020) proposed a span-level editing type that allowed to generate a span as insertion or replacement. Lewis (Reid and Zhong, 2021) used a two-step editor which first predicted coarse editing types and then filled in replacements and insertions. Edit5 (Mallinson et al., 2022) was a semi-auto-regressive approach with non-auto-regressive text labeling and auto-regressive decoding. It first decided the subset of input tokens to be retained using an encoder, then reordered the tokens with a pointer module, and finally infilled the missing tokens using a decoder. In this work, we proposed a simple but effective approach to address the copy issue by only decoding the changed sub-sentences.

3 Dataset Construction

We formulate our task as follows: given three consecutive sentences M , S , N , the model should output a polished sentence T that is more engaging than S while maintaining the original meaning of S and the coherence along with M and N . Since there are no available data for this task, we construct a new dataset through automatic annotation.

Firstly, we use an off-the-shelf OCR tool to col-

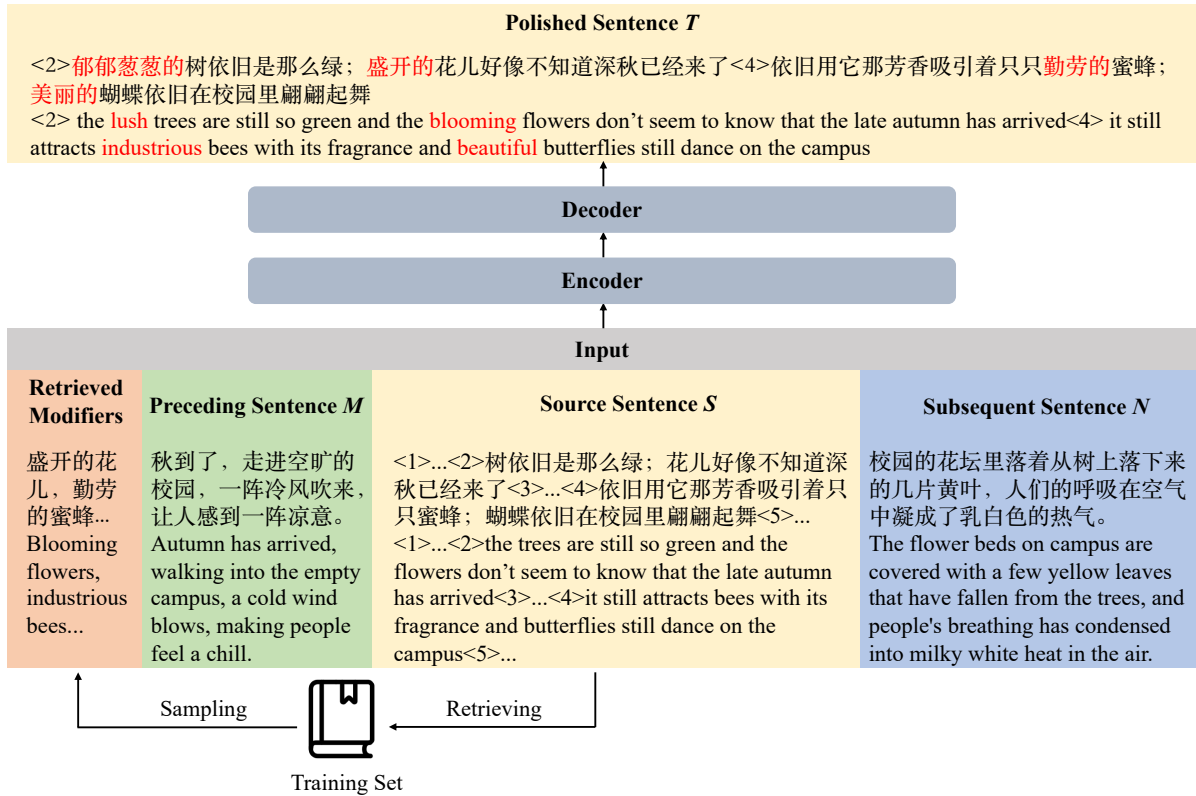


Figure 1: Model overview. We split the source sentence S into sub-sentences by commas and only generate the modified sub-sentences. We also retrieve relevant modifiers from the training set, which are taken as input. The modifiers added by the model are marked in red.

lect about 50k engaging sentences and a vocabulary of 61k engaging words from several books¹, and collect 26k high-quality Chinese student essays from public resources² that describe the scenery and thus potentially contain lots of engaging sentences. Then we take every triple of adjacent three sentences in these texts collected from books and websites as M , T and N ³, respectively. We obtain S by removing modifiers in T . Specifically, we adopt a public Chinese NLP toolkit LTP⁴ to identify attribute and adverbial words in T , and regard those words included in the vocabulary of engaging words as the modifiers that can be removed. Note that we also remove the structural particle words including “的” and “地” following the removed modifiers to ensure the fluency of the final sentence S . If there is no removed modifier in T , we will

¹ 《1000篇好词好句好段（初中）》，《小学生好词好句好段手册（新课标教材版）》，《书通网》，《黄冈作文-小学生好词好句好段》。

² <https://www.leleketang.com/zuowen/list10-0-0-1-1.shtml>

³ An engaging text example collected from books may contain less than three sentences. In this case, M or N can be missing.

⁴ <https://github.com/HIT-SCIR/ltp>

discard the example.

Table 2 shows the statistics of our dataset. Considering that the auto-constructed inputs may be different from real texts, we construct an additional test set, i.e., $\text{Test}_{\text{Real}}$, where the inputs are original sentences whose modifiers are not removed. We set the size of both test sets to 1000 to balance the estimation error and the inference time.

4 Methodology

An overview of our model is shown in Figure 1. We build our model on LongLM_{Large} (Guan et al., 2022) for the sentence polishing task, which is an encoder-decoder model pretrained on Chinese novels with 1 billion parameters. Considering the significant overlap between the source sentence S and the polished sentence T , we split S into several sub-sentences by commas and require the model to generate only the modified sub-sentences. We describe the detailed input-output format in §4.1. Moreover, we retrieve relevant modifiers from the training corpus, which are taken as input for both training and inference to help the model find suitable modifiers. We show the retrieval augmentation

algorithm in §4.2.

4.1 Input-Output Format

In our pilot experiments, we take the concatenation of M , S and N as input⁵ and train the model to minimize the log-likelihood of the whole target output T . We observe that the model tends to directly copy S as the generation result. We conjecture this is because most tokens in T overlap with S during training, making the model take the shortcut of copying instead of generating new tokens. To alleviate this issue, we split the source sentence S into several sub-sentences by commas (S without commas is not split), and train the model to decode only the modified sub-sentences. As shown in Figure 1, the source sentence S is split into 5 sub-sentences by commas, and the decoder only needs to decode the second and the fourth sub-sentence since the left three sub-sentences remain unchanged. This training strategy not only reduces the ratio of generated tokens that completely copy from the source inputs, but also improves the generation speed. Finally, we use the generated sub-sentences to replace the original ones in S to obtain the whole output sentence.

4.2 Retrieval Augmentation

We observe that the model trained with the framework described in §4.1 sometimes adds unsuitable modifiers (e.g., using “colourful” to modify “sun”). To alleviate this problem, we propose a retrieval augmentation algorithm to prompt the model to find suitable modifiers.

To this end, we first collect all pairs of modifiers and corresponding modified words from the engaging sentences in the training corpus, including attribute words paired with the modified nouns, and adverbial words paired with the modified verbs. We identify the attribute and adverbial words in each sentence through the dependency parsing toolkit of LTP. Furthermore, we restrict that the identified attribute and adverbial words are included in the vocabulary of engaging words. In this way, we obtain a dictionary that can map a noun or verb to a list of its suitable modifiers.

During inference, we first find all nouns and verbs in the source sentence S that are included in the dictionary and do not have engaging modifiers. Then, we randomly sample at most five modifiers for each noun or verb from its corresponding list

of modifiers. The sampled modifiers along with the nouns and verbs are inserted before the original input. **During training**, considering that conditioning on too many modifiers that are not used for generating may make the model tend not to use the retrieved modifiers, we drop the retrieved modifiers corresponding to the nouns or verbs that do not have any modifiers in the target output with a probability p_1 . Furthermore, to avoid excessive dependence on the retrieved modifiers, for those nouns or verbs that have modifiers in the target output, we drop all corresponding modifiers with a probability p_2 , randomly sample at most five modifiers with the probability p_3 , or randomly sample at most four modifiers along with the ground-truth modifier with a probability p_4 . Note that $p_2 + p_3 + p_4 = 1$. Finally, we insert the selected modifiers with the nouns and verbs before the original input.

5 Experiments

5.1 Compared Models

We compare our model with the following three variants: **(1) Retrieve**, which randomly samples modifiers from the retrieved phrases, and then adds the sampled modifiers to the sentence without using neural language model. **(2) LongLM_{vanilla}**, which generates the whole polished sentence instead of only generating the modified sub-sentences; **(3) LongLM_{no-retrieve}**, which generates the modified sub-sentences without retrieval augmentation.

5.2 Experiment Settings

We initialize our model using the pretrained checkpoint of LongLM_{Large}. For retrieval augmentation, the probability p_1, p_2, p_3, p_4 is set to 0.75, 0.25, 0.25, 0.5, respectively. And the sampled modifiers are dynamically changed at different epochs during training. We run our experiments on 6 Tesla V100 GPUs (32GB memory). We use DeepSpeed⁶ with mixed precision to train our model, which helps significantly reduce the memory usage. We set learning rate to 5e-5 and batch size per GPU to 8. The maximum input length is set to 384 and the maximum output length is set to 128. We train the model for 10 epochs and select the best checkpoint that has the lowest perplexity on the validation set. During inference, we combine beam search (beam size = 10) (Graves, 2012), top- k sampling ($k = 50$) (Fan et al., 2018) and top- p

⁵ M, S and N are separated by <sep>.

⁶<https://github.com/microsoft/DeepSpeed>

sampling ($p = 0.9$) (Holtzman et al., 2020) for decoding. We apply these settings to all models.

5.3 Automatic Evaluation

We evaluate the models on both $\text{Test}_{\text{Auto}}$ and $\text{Test}_{\text{Real}}$. We adopt the following two metrics: **(1) Copy ratio:** It calculates the ratio of samples whose output and input are exactly the same. **(2) # Added Modifiers:** It calculates the averaged number of added modifiers in the outputs. These two metrics aim to measure the differences between the inputs and outputs.

Models	Copy Ratio	# Added Modifiers
LongLM _{vanilla}	3.8% / 38.6%	1.16 / 0.40
LongLM _{no-retrieve}	0.2% / 8.5%	1.27 / 0.94
Ours	0.3% / 7.1%	1.23 / 0.92

Table 3: Automatic evaluation result. The two values separated by “/” indicate the performance on $\text{Test}_{\text{Auto}}$ and $\text{Test}_{\text{Real}}$, respectively.

The automatic evaluation result is shown in Table 3. We do not report the results of the Retrieve model because it adds as many modifiers as possible without considering fluency. LongLM_{vanilla} has the highest copy ratio and adds the fewest modifiers among the three models. Moreover, all three models tend to copy more from inputs on $\text{Test}_{\text{Real}}$ than $\text{Test}_{\text{Auto}}$, and LongLM_{vanilla} shows a larger margin than other two models which only generate the modified sub-sentences. The result suggests the worse generalization ability of LongLM_{vanilla}. Besides, we find that our model has a lower copy ratio than LongLM_{no-retrieve} when tested on $\text{Test}_{\text{Real}}$, indicating that the retrieval augmentation module helps improve generalization to real texts by providing references for adding modifiers.

Aspects	Scores	Descriptions
Fluency	0	The output is obviously not fluent.
	1	The output is a little bit not fluent.
	2	The output is fluent.
Correctness	0	The modifiers in the output are incorrect.
	1	The correctness of the modifiers in the output is ambiguous.
	2	The modifiers in the output are correct.
Engagingness	1	The Engagingness of the output drops.
	2	The engagingness of the output is unchanged.
	3	The engagingness of the output improves slightly.
	4	The engagingness of the output improves.
	5	The engagingness of the output improves significantly.

Table 4: Scoring rules in manual evaluation.

Models	Fluency (κ)	Correctness (κ)	Engagingness (κ)
Retrieve	0.82 (0.42)	0.61 (0.50)	2.00 (0.64)
LongLM _{vanilla}	1.93 (0.69)	1.87 (0.52)	2.91 (0.75)
LongLM _{no-retrieve}	1.81 (0.55)	1.73 (0.52)	3.21 (0.76)
Ours	1.88 (0.57)	1.84 (0.57)	3.44 (0.85)

Table 5: Manual evaluation result. We show Fleiss’s kappa value κ in the parentheses to measure the inter-annotator agreement.

5.4 Manual Evaluation

Considering there may be many plausible modifications for the same input, it is hard to automatically evaluate the quality of the added modifiers. Therefore, we resort to manual evaluation in terms of three aspects including: **(1) Fluency (0-2):** whether the polished sentence is fluent in terms of grammatical quality; **(2) Correctness (0-2):** whether the added modifiers in the polished sentence are suitable to modify the corresponding nouns, verbs, etc.; **(3) Engagingness (1-5):** whether the engagingness of the polished sentence improves compared with the source sentence. We show the detailed scoring rules in Table 4. We first randomly sample 100 inputs from $\text{Test}_{\text{Real}}$. Then we use the Retrieve model, LongLM_{vanilla}, LongLM_{no-retrieve} and our model to generate polished sentences for the sampled inputs. For each generated sample, we hire three well-trained professional annotators to give a score for each of the three evaluation aspects. Note that these aspects are evaluated independently. We directly average the scores given by three annotators to get the final scores.

Table 5 shows the evaluation results. All results show moderate or better ($\kappa > 0.4$) inter-annotator agreement. By comparing LongLM_{vanilla} and LongLM_{no-retrieve}, we can see that only generating the modified sub-sentences helps improve the engagingness of the polished sentence due to the lower copy ratio. However, the drop of copy ratio also brings a higher risk of adding unsuitable modifiers. Our retrieval augmentation algorithm improves the correctness of the polished sentences by providing multiple possible modifier candidates. Moreover, the suitable modifiers make the polished sentences more fluent and engaging. However, if we remove LongLM and only utilize the retrieved modifiers, it is hard to create fluent and coherent sentences as the result shows, which suggests the necessity to integrate the contextualization ability of generation models. In summary, our model

<i>M</i>	<i>S</i>	<i>N</i>	LongLM _{no-retrieve}	Ours
从远处看，桃花星星点点，似一群娇小可爱的小女孩。从近处看，这桃花恰似那闭月羞花的少女，在风中轻歌曼舞，柔美的身姿随风摇摆。(From a distance, the peach blossoms are dotted, like a group of petite and cute little girls. From a close point of view, this peach blossom is just like the girl who closes the moon and shy flowers, singing and dancing softly in the wind, and the feminine posture sways with the wind.)	冬天，桃花树养精蓄锐，吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开了，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)	我觉得不是“梅花香自苦寒来”，还是“桃花艳自苦寒来。”(I think it's not just "plum blossom fragrance comes from bitter cold", but also "peach blossoms come from bitter cold".)	冬天，桃花树养精蓄锐， 无助地 吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开了，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, helplessly sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)	冬天，桃花树养精蓄锐， 贪婪地 吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的 沉闷 的惊雷；终于，明媚的春天来了，当 热情的 迎春花开过，它们便 骄傲地 向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, greedily sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the cold winter. Finally, the bright spring has come, and when the enthusiastic spring flowers have bloomed, they proudly show the world their beautiful posture.) Retrieved Phrases: 沉闷的惊雷, 贪婪地吮吸, 热情的迎春花, 充分地展示, 火辣辣的阳光... (Dull thunder, greedily suck, enthusiastic spring flowers, fully show, fiery sunlight...)
秋风一阵阵吹来，一层层桔色的“海浪”迎面“扑”来，感觉像看四维电影一样。(The autumn wind blows in waves, and the layers of orange "waves" "flutter" in the face, feeling like watching a four-dimensional movie.)	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the petals, the largest of which is the size of Daddy's fist.)	花是桔红色的，绿色的叶子把花瓣裹住。(The flowers are orange-red, and the green leaves wrap the petals.)	Same as the input source sentence <i>S</i> .	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被 鲜艳的 花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the brightly colored petals, the largest of which is the size of Daddy's fist.) Retrieved Phrases: 鲜艳的花瓣, 纤细的花蕊... (brightly colored petals, slender flower buds...)

Table 6: Cases generated by different models on Test_{Real}. *M*, *S* and *N* are the preceding, source and subsequent sentences, respectively. LongLM_{vanilla} copies the source sentences for both cases and we omit the generation results. We mark the added modifiers generated by LongLM_{no-retrieve} in orange. In the generation result of ours, we mark the added modifiers that have been retrieved in red, and others in blue.

can improve the engagingness significantly⁷ while keeping fluency and correctness comparable with baselines.

5.5 Case Study

We show two cases in Table 6. Our model can add suitable modifiers in multiple sub-sentences with the help of various retrieved modifiers. In contrast, LongLM_{no-retrieve} uses “helplessly” to modify “sucking”, which is reasonable in isolation but is incoherent with the context, thus decreasing the engagingness of the sentence. We additionally show the result of the Retrieve model in Table 7 in the appendix.

6 Demonstration

We have deployed our model online to automatically polish the source sentence given its context. Figure 2 shows a screenshot of our demo website. Users need to enter the source sentence and its preceding and subsequent sentences. Note that the

⁷ $p < 0.01$ when compared with LongLM_{vanilla} (Wilcoxon signed-rank test).



Figure 2: A screenshot of our demo website.

source sentence is mandatory but its context can be empty. Then users can submit the request and the result will be returned after a few seconds. We show the polished sentence at the bottom of the page, and the retrieved modifiers for reference.

7 Conclusion

We propose a new task named sentence polishing, which requires polishing a given sentence while

maintaining fluency and coherence with the context. To this end, we construct about 160k parallel examples by removing modifiers in collected engaging sentences. Then we fine-tune LongLM to reconstruct the original sentences from the corrupted ones by generating the modified sub-sentences. We also propose a retrieval augmentation algorithm to retrieve engaging modifiers from the training set, which can help generate suitable modifiers. Automatic and manual evaluation demonstrate strong performance of our model to generate engaging sentences. We have deployed our model online for public use. Although we focus on adding modifiers in this paper, the perturbation-and-reconstruction framework can be potentially adapted to other polishing techniques such as adding metaphors, which is left as future work. Moreover, although we train our model on collected Chinese data, we believe the method can be easily transferred to other languages.

References

- Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7375–7382.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.
- Cristina Garbacea and Qiaozhu Mei. 2022. Why is constrained neural language generation particularly challenging? *arXiv preprint arXiv:2206.05395*.
- Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.
- Jian Guan, Zhuoer Feng, Yamei Chen, Ruilin He, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2022. Lot: A story-centric benchmark for evaluating chinese long text understanding and generation. *Transactions of the Association for Computational Linguistics*, 10:434–451.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. [The curious case of neural text de-generation](#). In *International Conference on Learning Representations*.
- 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.
- Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020a. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840.
- Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020b. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663.
- Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Edit5: Semi-autoregressive text-editing with t5 warm-start. *arXiv preprint arXiv:2205.12209*.
- Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. Felix: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255.
- Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. 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 5054–5065.
- Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanyski. 2020. [GECToR – grammatical error correction: Tag, not rewrite](#). In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.
- Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.
- Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295.

- Machel Reid and Victor Zhong. 2021. Lewis: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944.
- Felix Stahlberg and Shankar Kumar. 2020. Seq2edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159.
- 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.
- 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.
- Stephen P Witte and Lester Faigley. 1981. Coherence, cohesion, and writing quality. *College composition and communication*, 32(2):189–204.
- Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. 2020. A survey of deep learning techniques for neural machine translation. *arXiv preprint arXiv:2002.07526*.

A Case Study

We additionally show two cases of the Retrieve model in Table 7. We can see that the Retrieve model can add unsuitable modifiers such as *light-some* or duplicated modifiers (e.g., "一团团" after "一大团"). Moreover, the sentence generated by Retrieve is less fluent than the sentence generated by our model.

<i>M</i>	<i>S</i>	<i>N</i>	Retrieve	Ours
<p>从远处看，桃花星星点点，似一群娇小可爱的小女孩，从近处看，这桃花恰似那闭月羞花的少女，在风中轻歌曼舞，柔美的身姿随风摇摆。(From a distance, the peach blossoms are dotted, like a group of petite and cute little girls. From a close point of view, this peach blossom is just like the girl who closes the moon and shy flowers, singing and dancing softly in the wind, and the feminine posture sways with the wind.)</p>	<p>冬天，桃花树养精蓄锐，吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开过，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)</p>	<p>我觉得不是“梅花香自苦寒来”，还是“桃花艳自苦寒来。”(I think it's not just "plum blossom fragrance comes from bitter cold", but also "peach blossoms come from bitter cold".)</p>	<p>冬天，桃花树养精蓄锐，尽情吮吸着冬天仅有的丝丝清澈的阳光，它们饱经风霜，静待那一声撕破凛冽的寒冬的沉闷的惊雷；终于，明媚的春天轻盈地来了，当许许多多的迎春花热热闹闹开过，它们便向世人努力展示出那闭月羞花、沉鱼落雁的轻盈的一种姿态。(In winter, peach blossom trees nourish themselves, enjoyably sucking in the only shred of clear sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the nippy and cold winter. Finally, the bright spring has come airily, and when a lot of spring flowers have bloomed with high spirits, they show the world their beautiful and light-some posture.)</p>	<p>冬天，桃花树养精蓄锐，贪婪地吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的沉闷的惊雷；终于，明媚的春天来了，当热情的迎春花开过，它们便骄傲地向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, greedily sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the cold winter. Finally, the bright spring has come, and when the enthusiastic spring flowers have bloomed, they proudly show the world their beautiful posture.)</p> <p>Retrieved Phrases: 沉闷的惊雷，贪婪地吮吸，热情的迎春花，充分地展示，火辣辣的阳光... (Dull thunder, greedily suck, enthusiastic spring flowers, fully show, fiery sunlight...)</p>
<p>秋风一阵阵吹来，一层层桔色的“海浪”迎面“扑”来，感觉像看四维电影一样。(The autumn wind blows in waves, and the layers of orange "waves" flutter in the face, feeling like watching a four-dimensional movie.)</p>	<p>仔细一看，像一大团桔红色的火焰在燃烧，花蕊被花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the petals, the largest of which is the size of Daddy's fist.)</p>	<p>花是桔红色的，绿色的叶子把花瓣裹住。(The flowers are orange-red, and the green leaves wrap the petals.)</p>	<p>仔细一看，像一大团桔红色的粗犷的火焰在一团团燃烧，漂亮的花蕊被娇嫩的小花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large and clouds of orange-red and rough flame burning, and the beautiful flower buds are tightly held by the delicate petals, the largest of which is the size of Daddy's fist.)</p>	<p>仔细一看，像一大团桔红色的火焰在燃烧，花蕊被鲜艳的花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the brightly colored petals, the largest of which is the size of Daddy's fist.)</p> <p>Retrieved Phrases: 鲜艳的花瓣，纤细的花蕊... (brightly colored petals, slender flower buds...)</p>

Table 7: Cases generated by the Retrieve model on the same test examples as Table 6. We mark the added modifiers generated by the Retrieve model in orange. We also show the generation result of our model for reference.

Effdit: An Assistant for Improving Writing Efficiency

Shuming Shi, Enbo Zhao, Wei Bi, Deng Cai, Leyang Cui, Xinting Huang, Haiyun Jiang, Duyu Tang, Kaiqiang Song, Longyue Wang, Chengyan Huang, Guoping Huang, Yan Wang, Piji Li
ailabnlp@tencent.com

Abstract

Writing assistants are valuable tools that can help writers improve their writing skills. We introduce Effdit (**Efficient and Intelligent Editing**), a digital writing assistant that facilitates users to write higher-quality text more efficiently through the use of Artificial Intelligence (AI) and Natural Language Processing (NLP) technologies. We significantly expand the capacities of a writing assistant by providing functions in three modules: text completion, hint recommendation, and writing refinement. Based on the above efforts, Effdit can efficiently assist users in creating their own text. Effdit has been deployed to several Tencent products and publicly released at <https://effdit.qq.com/>.

1 Introduction

Effective communication through writing is crucial in modern society, as it allows individuals to share their thoughts, opinions, and responses, thereby enabling them to establish meaningful connections with others. However, writing could be a challenging task, regardless of whether you are a beginner or an experienced writer. It requires a combination of creativity, organization, and critical thinking. To address this challenge, several writing assistant tools ([Grammarly](#); [QuillBot](#)) have been developed to help users write faster and more efficiently. These tools use Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques to provide suggestions, corrections, and recommendations to improve writing quality.

In general, the writing process can be divided into three main stages: generating and proposing ideas, composing coherent sentences, and refining the language through editing and polishing ([Flower and Hayes, 1980](#)). Tracking this, we present Effdit (**Efficient and Intelligent Editing**) to provide users with an innovative writing experience, which significantly expands the capacities of a typical writing assistant by providing three categories of functions:

- **Text Completion** module intends to provide high-quality continuations for any given textual prefix for improving the user’s writing efficiency. Effdit offers both short-form and long-form text completion. For short-form completion, Effdit provides two functions including Phrase Completion which takes the prefix as input and suggests suitable follow-up phrases, and a Cloud Input Method Editor (IME) which can pop up instant suggestions during typing. Cloud Chinese Pinyin and English input methods are provided. Meanwhile, Sentence Completion automatically completes the whole sentence based on the prefix, serving as a long-form assistance. The completion can be done by either retrieving from an existing datastore or generating using advanced neural language models. To help users create fun and distinctive content, Effdit also supports stylistic automatic writing for Chinese novel writing. Specifically, four distinct and popular novel styles are covered currently.

- **Hint Recommendation** module can help users to brainstorm when they are struggling to find the inspiring words or are dealing with writer’s block. This module can recommend various hints given the input keywords, such as mono-lingual and multi-lingual retrieval example sentences, generated sentences by infilling the input keywords, representative papers, and a super-power dictionary that can suggest synonyms, antonyms, related words, possible modifiers, and related entities.

- **Writing Refinement** module is designed to help users refine texts they have already typed down by suggesting improvements in grammar, syntax, phrase and vocabulary. Grammar Error Correction (GEC) focuses on identifying and correcting grammar mistakes inside the sentence. Effdit additionally provides grammar-aware explanations and evidence words for corrections in English GEC. Effdit provides three different levels of text polishing: phrase polishing, sentence rewriting (i.e. para-

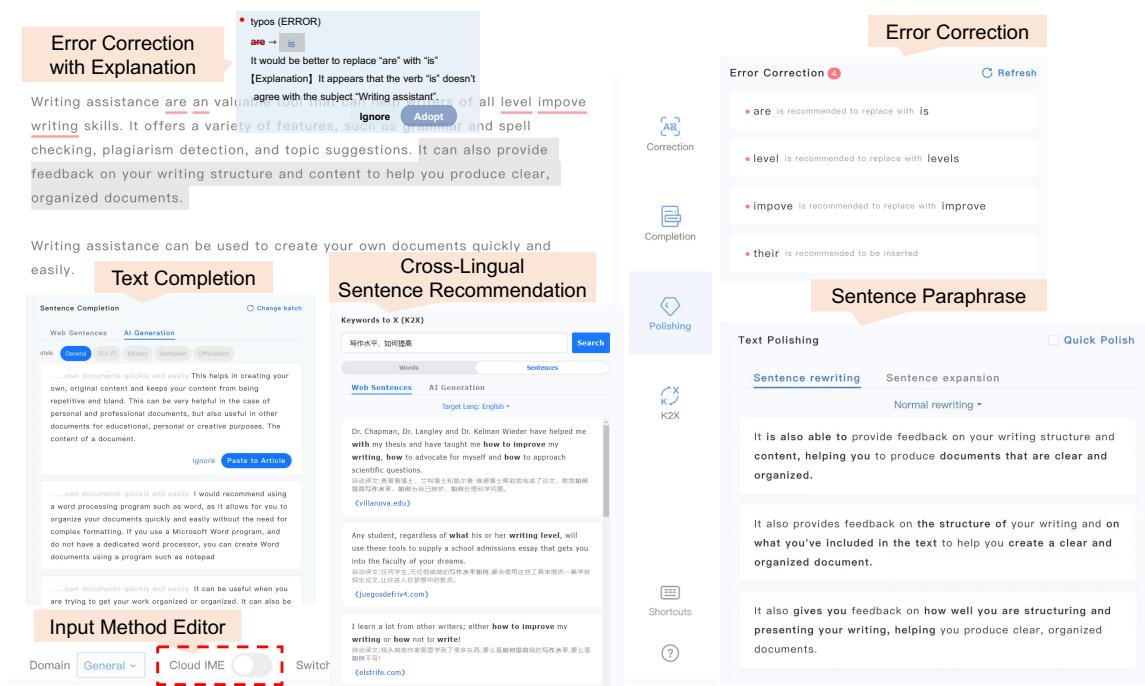


Figure 1: Screenshot of the Effdit online demo. To save our space, we put the results of Text Completion (Sentence Generation) and Hint Recommendation (Cross-Lingual Sentence Recommendation) on the bottom of the left part, which are originally on the right pane.

phrasing) and sentence expansion. By selecting some words, phrase-level polishing automatically recommends words that are more suitable for the context, making the overall sentence more accurate and vivid. Sentence rewriting automatically modifies a sentence while retaining its semantics, thus enhancing sentence diversity. Sentence expansion generates longer sentences with consistent but richer semantics for a target sentence by adding some modifiers.

The first version of Effdit was released in April, 2022. Tencent AI lab constantly maintains and updates Effdit, aiming to provide users with a better writing experience. The current version supports both Chinese and English. Effdit is now providing support for many products and users both inside and outside Tencent. Detailed documents and tutorials are available at <https://effdit.qq.com/en>.

2 Overview

Figure 1 illustrates the process of users using the online demo of Effdit. The left part is a plain-text editor for users to add and edit text, whereas the right pane is for triggering most core functions of Effdit and displaying corresponding results. In addition, some UI elements are at the bottom for

changing domains and setting up the cloud IME. Table 1 presents a comparison between Effdit and several well-designed writing assistants. As can be seen, previous writing assistants typically provide the function of error checking (to detect and correct spelling and grammatical errors) and limited text-rewriting functionality. In Effdit, we significantly expand the capacities of a writing assistant by providing functions in three perspectives. More technical details can be found at Shi et al. (2022).

2.1 Architecture

Figure 2 shows three modules supported by Effdit: Text Completion module takes the incomplete text as input, and help users compose coherent sentences; Hint Recommendation module suggests relevant words/phrases/sentences given the input words; Writing Refinement module polishes the existing passage to improve writing quality. These modules are introduced in the following subsections.

3 Text Completion Module

Effdit offers both short-form and long-form assistance for text completion, which can help users enhance the efficiency and effectiveness of their writing. As an advanced feature, Effdit can provide completion suggestions in specific writing styles.

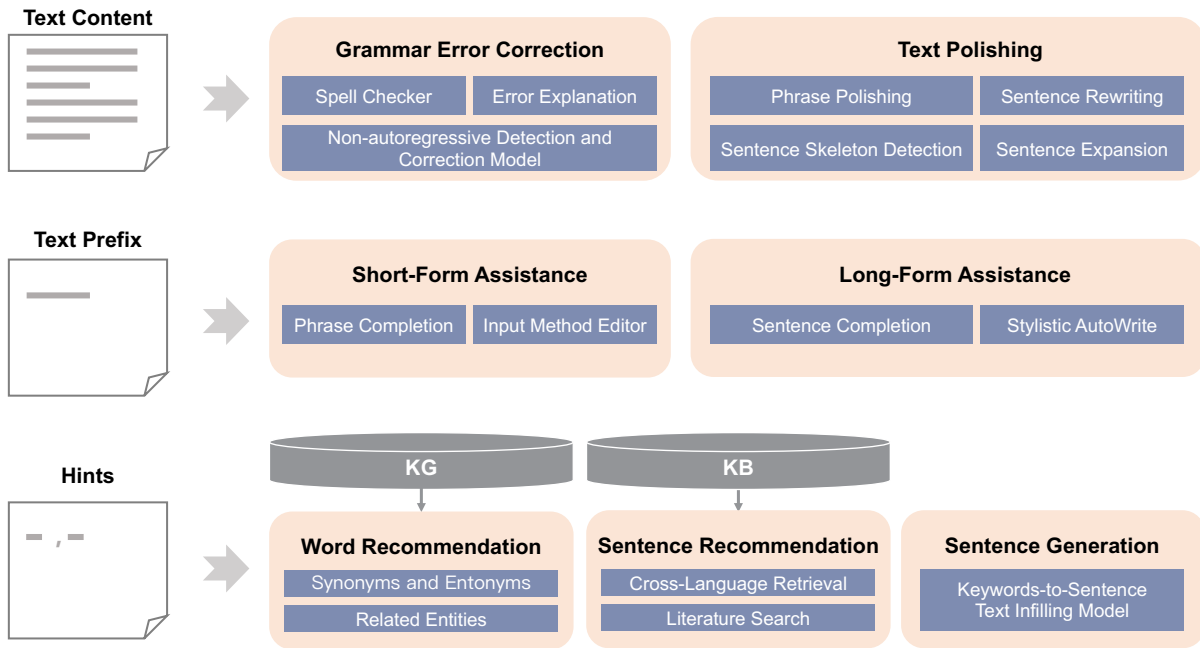


Figure 2: Overview of all modules and the main backbone models in Effdit. We categorize them based on the semantic completeness of the model inputs: Text Content refers to all text that users have written down, which could include multiple sentences, paragraphs, and articles; Text Prefix is a short piece of text with incomplete or partial semantics that should be continued; Keywords are one or a few words that users want to explore further information as hints to inspire their writing.



Figure 3: Phrase completion examples (Left: English; Right: Chinese).

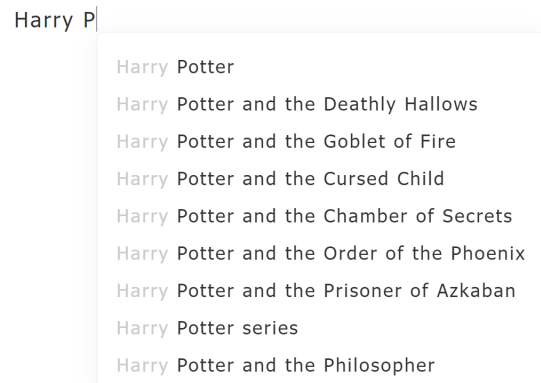


Figure 4: An example of Cloud Input Methods.

Currently, four unique styles are available, including science fiction (SCI-FI), military novel (Military), martial art fantasy (MartialArt), and urban story (Urban), allowing users to create engaging and stylistic content that is both fun and distinctive.

Phrase completion Figure 3 shows an example. It is noteworthy that Effdit considers both the prefix and suffix for phrase completion, setting it apart from most other writing assistants that only take the prefix into account. As illustrated in the left part of the figure, when the caret is after the first letter “B”, the top completion results contain both the prefix “B” and the suffix “Spears”. Phrase comple-

tion can improve writing efficiency and reduce the chance of typographical errors (Lee et al., 2021). For example, if a user types “Los A” and triggers phrase completion, the topmost suggestion is “Los Angeles”, which can be directly selected by the user. In this way, potential spelling mistakes such as “Los Angelas” can be avoided.

Cloud input method editors (Cloud IMEs) Effdit provides cloud IMEs for both Chinese and English to improve the input efficiency of words and phrases. Instant suggestions pop up when the user is typing in our text editor. Figure 4 depicts an example where the names of some novels in the

Function	QuillBot	Grammarly	Deepl	Pitaya	Phrasebank	Effdit
Phrase Completion	-	-	-	-	-	✓
Retrieval-based Completion	-	-	-	-	-	✓
Generation-based Completion	-	-	-	-	-	✓
Mono-lingual Retrieval	-	-	-	✓	-	✓
Cross-lingual Retrieval	-	-	-	-	-	✓
keyword-to-sentence	-	-	-	✓	-	✓
dictionary	-	-	-	✓	✓	✓
Grammar Error Correction	✓	✓	✓	✓	-	✓
Phrase Polish	✓	✓	✓	✓	-	✓
Sentence Rewrite	✓	-	-	-	-	✓
Sentence Expansion	-	-	-	✓	-	✓
Supported language	En	En	En	En&Zh	En	En&Zh

Table 1: Comparison between Effdit and other writing assistants.

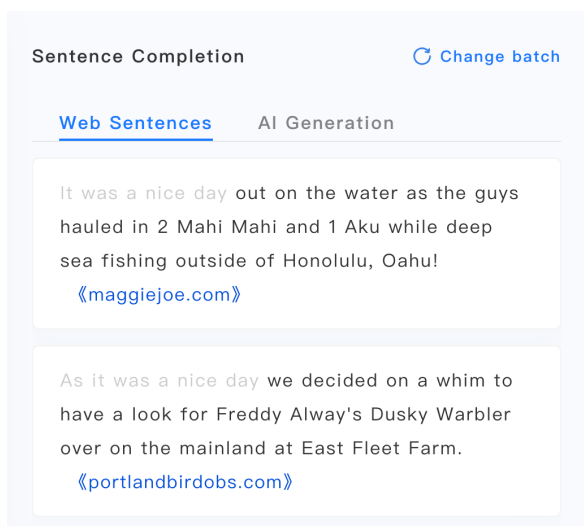


Figure 5: Examples of sentence-level completion using Web Retrieval. The input prefix is “It was a nice day”.

Harry Potter series are shown as suggestions.

Sentence completion Effdit also provides sentence completion suggestions (Zweig et al., 2012). Figure 5 and 6 demonstrate the results of two completion methods: web retrieval and AI generation. Similar to phrase completion, Effdit supports both Chinese and English. The web retrieval method utilizes a large collection of high-quality sentences to search for the most similar sentences to the input prefix. On the other hand, the AI generation method leverages large language models to generate the most probable continuations. To alleviate the serious degeneration problem, i.e., the generated texts from the language model tend to be dull and contain undesirable repetitions at different lev-

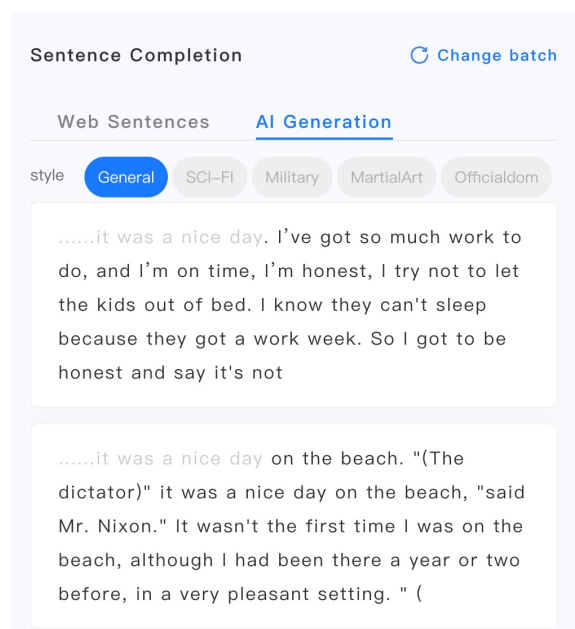


Figure 6: Examples of sentence-level completion using AI generations. The input prefix is “It was a nice day”.

els (e.g., token-, phrase-, and sentence-level), we apply the contrastive search (Su et al., 2022) and FSD decoding (Yang et al., 2023) methods. The key intuitions behind are: (i) At each decoding step, the output should be selected from the set of most probable candidates predicted by the model to better maintain the semantic coherence. (ii) The patterns that occur in the input prefix and/or the previously generated sequence should be penalized to avoid degeneration. Effdit integrates both retrieval and generation methods by presenting multiple candidates from each of them, enabling users to choose the best option according to their preferences.

Stylistic AutoWrite To assist users in crafting engaging, creative, and versatile stories, Effdit expands its sentence completion function to include stylistic autowrite (Dathathri et al., 2020). With four novel writing styles to choose from (SCI-FI, Military, MartialArt, and Urban), Effdit generates multiple possible continuations that adhere to the selected writing style and the input prefix. This functionality allows users to explore multiple stylistic possibilities and inspires their creativity. Unlike other text completion modules, stylistic autowrite only supports Chinese writing currently, and we are trying to upgrade our models for English and more languages, which will be released soon.

4 Hint Recommendation Module

Effdit can take one or more keywords as input and recommend a list of related words, sentences and documents. Especially, each recommended output sentence/document either contains the input keywords, or is semantically related to them. These modules are built for the scenario that users only have some basic concepts in their minds but do not have enough information to organize sentences to express their ideas.

Word Recommendation Effdit provides a super-power dictionary for users to first explore a wide range of related words including synonyms, antonyms, similar words, possible modifiers, and related entities, etc. As shown in Figure 7, given a keyword “linguistic”, Effdit produces both monolingual and bi-lingual dictionaries.

Keyword2sentence (K2S) This module takes several keywords as input and returns a list of sentences. Effdit supports the retrieval-based method and the generation-based method. For the retrieval-based method, top sentences containing the input keywords are retrieved from a corpus of high-quality sentences collected beforehand from the Web. Generation-based K2S is formulated as a text-infilling problem (Zhu et al., 2019; Donahue et al., 2020), which is to generate missing spans of text. Our system considers a general text-infilling setting, where the incomplete sentence can contain an arbitrary number of blanks to be filled in, and each blank can involve an unknown number of tokens. Illustrated examples are provided in Figure 8. For both types of results, Effdit displays multiple candidates, from which the user can select the most appropriate one for reference.



Figure 7: Examples of word recommendations provided by the super-power dictionary (Upper: Mono-lingual; Lower: Bilingual).

Sentence/Document Recommendation We use information retrieval methods (Baeza-Yates et al., 1999; Manning, 2008) to index a large corpus of sentences and documents. Note that Effdit supports cross-lingual retrieval, where the output is English sentences and the input can be a mixture of Chinese and English keywords. Effdit also support document-level search, especially paper recommendation. Unlike other paper search services such as (Scholar), the paper search of Effdit focuses more on recalling papers that are semantically related to the input keywords other than those containing the input keywords in their titles or contents. In particular, when the input keyword is a research topic, important papers on this topic will be returned even if some keywords are missing in some results.

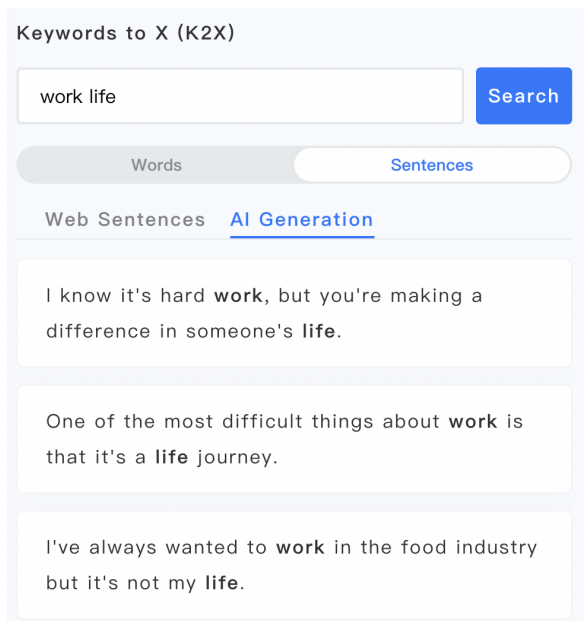


Figure 8: Examples of keywords2sentence.

5 Writing Refinement Module

Grammar Error Correction The grammar error correction (GEC) capability is a pivotal attribute for writers, particularly for novices and those utilizing a non-native language (Dahlmeier and Ng, 2012; Wang et al., 2020). The GEC module of Effdit can automatically detect and correct grammatical, syntactical, and spelling errors in written texts. As shown in the upper right part of Figure 1, Effdit offers users various types of suggestions, such as substitution, deletion, and insertion, during the process of error correction. We have a tail-to-tail model (Li and Shi, 2021) to handle substitution errors and a tailored pretrained model (Zhou et al., 2022) to handle deletion and insertion errors. Moreover, Effdit advances its GEC capabilities by providing grammar-aware explanations and evidence words. Specifically, as shown in the upper left part of Figure 1, the explanation describes why a correction was made to a particular error (e.g. the grammatical rule that was violated), and evidence words provide a clear indication of such error (Fei et al., 2023). With explainable GEC functions, Effdit can increase users' trust and acceptance of the corrections made by the algorithm, and help them to improve their writing skills via detailed and specific feedback.

Text Polishing Text polishing is a process of refining and improving written text by enhancing its clarity, readability, and overall quality (Bhagat and

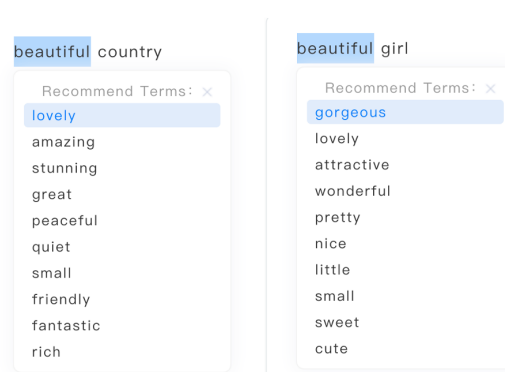


Figure 9: Examples of phrasal paraphrasing.

Hovy, 2013; Gupta et al., 2018). Effdit provides two functions, phrasal and sentential paraphrasing, to achieve the goal of text polishing. Phrasal paraphrasing involves rephrasing individual words or phrases within a sentence to create more impactful or clearer language. As shown in Figure 9, by phrasal paraphrasing, writers can avoid repetitive language and create a more interesting and varied writing style. Sentence paraphrasing, on the other hand, involves rewriting an entire sentence to convey the same meaning in a different way. The lower right quadrant of Figure 1 illustrates an instance in which a range of diverse paraphrased options are presented. Based on options suggested by Effdit, writers can then choose alternative phrasing that is more natural and intuitive. In this way, it would be able to eliminate awkward or convoluted sentence structures that make writing difficult to follow. Apart from normal rewriting, we also provide the conversion between classical↔modern Chinese language.¹ Different from modern language, classical language is often used in poetry, prose, and other forms of traditional literature. The modern→classical conversion can help writers seamlessly integrate classical elements into their writing, making their works more artistic and expressive. On the other hand, classical→modern conversion can accessibly help users better understand classical literature, especially in the field of education.

Text Expansion Effdit provides the function of text expansion by adding some elegant modifiers to an input sentence to make a longer one with rich information. In general, the new sentence will keep the core meaning with the input sentence. For ex-

¹https://en.wikipedia.org/wiki/Chinese_classics.

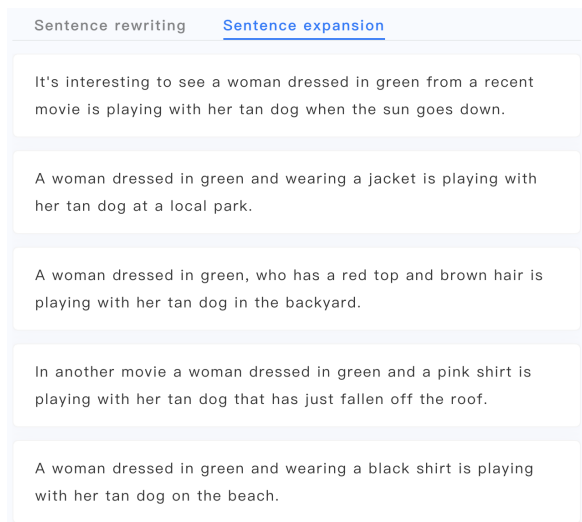


Figure 10: Examples of text expansion.

ample, given the text “A woman dressed in green is playing with her tan dog”, the expanded sentences are presented in Figure 10. Text expansion module was implemented by two complementary ideas. The first is to resort to syntactic parsing to extract the skeleton of a long sentence to build the training pairs, i.e., <skeleton, sentence>. Then the skeleton is used to generate the long sentence with a seq2seq model (Lewis et al., 2019). The second idea is to identify some places (mainly around a noun or verb) explicitly and then use a text in-filing model (Raffel et al., 2020) to predict some words or phrases that can be added to each place.

6 Conclusion

We introduced Effdit, a writing assistant that facilitates users to write high quality text efficiently using AI technologies. Our system supports two languages, Chinese and English, and has three categories of functions: text completion, hint recommendation, and writing refinement. With these modules, Effdit significantly expands the capacities of a typical writing assistant. In the future, we plan to keep improving the quality of each module to make the system more helpful and easy-to-use.

References

- Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.
- Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations*,.

DeepL. <https://www.deepl.com/write>.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501.

Yuejiao Fei, Leyang Cui, Sen Yang, Wai Lam, Zhenzhong Lan, and Shuming Shi. 2023. Enhancing grammatical error correction systems with explanations. Toronto, Canada. Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics.

Linda S. Flower and J. R. Hayes. 1980. The cognition of discovery: Defining a rhetorical problem. *College Composition and Communication*, 31:21–32.

Grammarly. <https://www.grammarly.com/>.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Dong-Ho Lee, Zhiqiang Hu, and Roy Ka-Wei Lee. 2021. Improving text auto-completion with next phrase prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4434–4438, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*.

Piji Li and Shuming Shi. 2021. Tail-to-tail non-autoregressive sequence prediction for chinese grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4973–4984.

Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.

Academic Phrasebank. <https://www.ref-n-write.com/academic-phrasebank/>.

Pitaya. <https://www.mypitaya.com/en>.

QuillBot. <https://quillbot.com/>.

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.

Google Scholar. <https://scholar.google.com/>.

Shuming Shi, Enbo Zhao, Duyu Tang, Yan Wang, Piji Li, Wei Bi, Haiyun Jiang, Guoping Huang, Leyang Cui, Xinting Huang, et al. 2022. Effidit: Your ai writing assistant. *arXiv preprint arXiv:2208.01815*.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. In *Advances in Neural Information Processing Systems*.

Yu Wang, Yuelin Wang, Jie Liu, and Zhuo Liu. 2020. A comprehensive survey of grammar error correction. *arXiv preprint arXiv:2005.06600*.

Haoran Yang, Deng Cai, Huayang Li, Wei Bi, Wai Lam, and Shuming Shi. 2023. A frustratingly simple decoding method for neural text generation. *arXiv preprint arXiv:2305.12675*.

Cong Zhou, Yong Dai, Duyu Tang, Enbo Zhao, Zhangyin Feng, Li Kuang, and Shuming Shi. 2022. Pretraining chinese bert for detecting word insertion and deletion errors. *arXiv preprint arXiv:2204.12052*.

Wanrong Zhu, Zhiting Hu, and Eric Xing. 2019. Text infilling. *arXiv preprint arXiv:1901.00158*.

Geoffrey Zweig, John C Platt, Christopher Meek, Christopher JC Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 601–610.

WIZMAP: Scalable Interactive Visualization for Exploring Large Machine Learning Embeddings

Zijie J. Wang
Georgia Tech
jayw@gatech.edu

Fred Hohman
Apple
fredhohman@apple.com

Duen Horng Chau
Georgia Tech
polo@gatech.edu

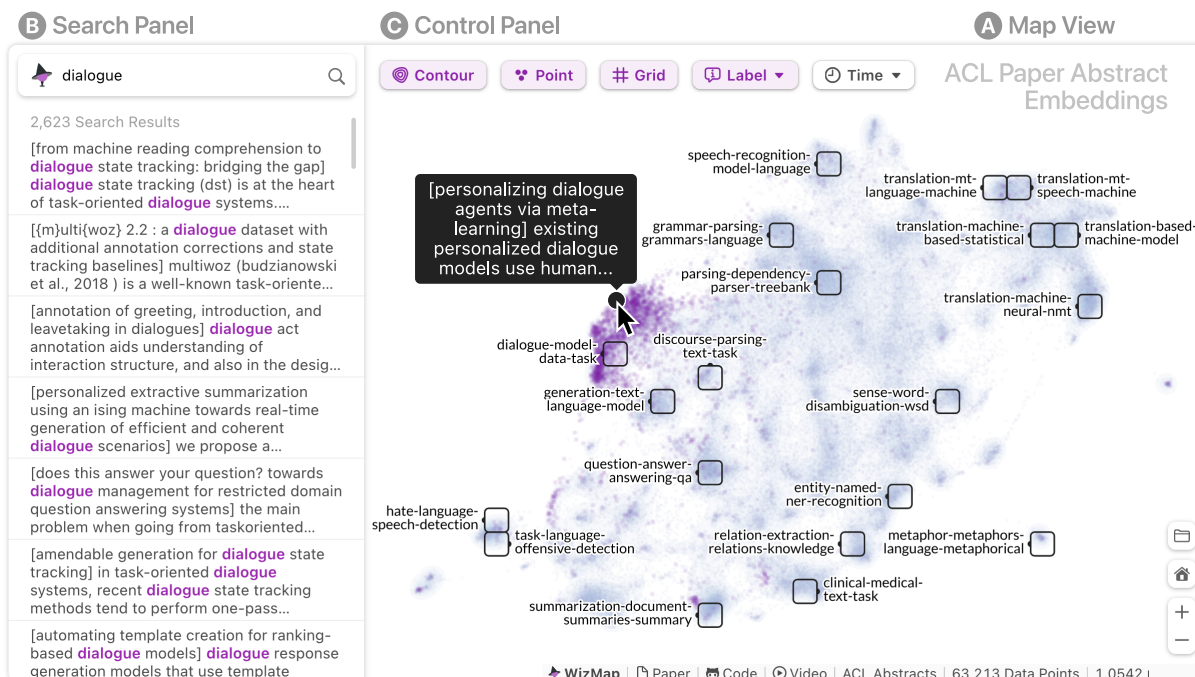


Fig. 1: WIZMAP empowers machine learning researchers and domain experts to easily explore and interpret millions of embedding vectors across different levels of granularity. Consider the task of investigating the embeddings of all 63k natural language processing paper abstracts indexed in ACL Anthology from 1980 to 2022. (A) The Map View tightly integrates a contour layer, a scatter plot, and automatically-generated multi-resolution embedding summaries to help users navigate through the large embedding space. (B) The Search Panel enables users to rapidly test their hypotheses through fast full-text embedding search. (C) The Control Panel allows users to customize embedding visualizations, compare multiple embedding groups, and observe how embeddings evolve over time.

Abstract

Machine learning models often learn latent embedding representations that capture the domain semantics of their training data. These embedding representations are valuable for interpreting trained models, building new models, and analyzing new datasets. However, interpreting and using embeddings can be challenging due to their opaqueness, high dimensionality, and the large size of modern datasets. To tackle these challenges, we present WIZMAP, an interactive visualization tool to help researchers and practitioners easily explore large embeddings. With a novel multi-resolution embedding summarization method and a familiar map-like interaction design, WIZMAP enables users to navigate and interpret embedding spaces with ease. Leveraging modern web technolo-

gies such as WebGL and Web Workers, WIZMAP scales to millions of embedding points directly in users' web browsers and computational notebooks without the need for dedicated backend servers. WIZMAP is open-source and available at the following public demo link: <https://poloclub.github.io/wizmap>.

1 Introduction

Modern machine learning (ML) models learn high-dimensional embedding representations to capture the domain semantics and relationships in the training data (Raghu et al., 2019). ML researchers and domain experts are increasingly using expressive embedding representations to interpret trained models (Park et al., 2022), develop models for new domains (Lee et al., 2018) and modalities (Ben-

younes et al., 2019), as well as analyze and synthesize new datasets (Kern et al., 2016). However, it can be difficult to interpret and use embeddings in practice, as these high-dimensional representations are often opaque, complex, and can contain unpredictable structures (Bolukbasi et al., 2016). Furthermore, analysts face scalability challenges as large datasets can require them to study millions of embeddings holistically (Tang et al., 2016).

To tackle these challenges, researchers have proposed several interactive visualization tools to help users explore embedding spaces (e.g., Smilkov et al., 2016; Liu et al., 2019). These tools often visualize embeddings in a low-dimensional scatter plot where users can browse, filter, and compare embedding points. However, for large datasets, it is taxing or even implausible to inspect embedded data point by point to make sense of the *global structure* of an embedding space. Alternatively, recent research explores using contour plots to summarize embeddings (Sevastjanova et al., 2022; Robertson et al., 2023). Although contour abstractions enable users to obtain an overview of the embedding space and compare multiple embeddings through superposition, a user study reveals that contour plots restrict users’ exploration of an embedding’s *local structures*, where users would prefer to have more visual context (Robertson et al., 2023). To bridge this critical gap between two visualization approaches and provide users with a holistic view, we design and develop WIZMAP (Fig. 1). Our work makes the following **major contributions**:

- **WIZMAP, a scalable interactive visualization tool** that empowers ML researchers and domain experts to explore and interpret embeddings with *millions* of points. Our tool employs a familiar map-like interaction design and fluidly presents adaptive visual summaries of embeddings across different levels of granularity (Fig. 2, § 4).
- **Novel and efficient method to generate multi-resolution embedding summaries.** To automatically summarize embedding neighborhoods with different degrees of granularity, we construct a quadtree (Finkel and Bentley, 1974) from embedding points and extract keywords (text data) or exemplar points (other data types) from tree nodes with efficient branch aggregation (§ 3).
- **An open-source¹ and web-based implementation** that lowers the barrier to interpreting

¹WIZMAP code: <https://github.com/poloclub/wizmap>

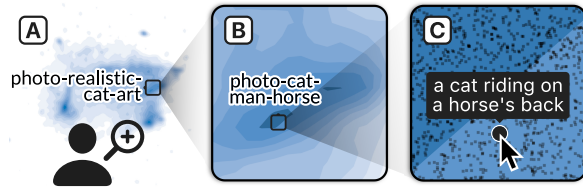


Fig. 2: WIZMAP enables users to explore embeddings at different levels of detail. (A) The contour plot with automatically-generated embedding summaries provides an overview. (B) Embedding summaries adjust in resolution as users zoom in. (C) The scatter plot enables the investigation of individual embeddings.

and using embeddings. We develop WIZMAP with modern web technologies such as WebGL and Web Workers so that anyone can access the tool directly in both their web browsers and computational notebooks without a need for dedicated backend servers (§ 4.4). For a demo video of WIZMAP, visit <https://youtu.be/8fJG87QVceQ>.

2 Background and Related Work

Researchers can extract a data point’s embeddings by collecting its corresponding layer activations in neural networks trained for specific tasks such as classification and generation (Raghu et al., 2019). Additionally, researchers have developed task-agnostic models, such as word2vec (Mikolov et al., 2013), ELMo (Peters et al., 2018), and CLIP (Radford et al., 2021) that generate transferable embeddings directly. These embeddings have been shown to outperform task-specific, state-of-the-art models in downstream tasks (Radford et al., 2021; Dwivedi et al., 2021).

2.1 Dimensionality Reduction

Embeddings are often high-dimensional, such as 300-dimensions for word2vec, or 768-dimensions for CLIP and BERT Base (Devlin et al., 2018). Therefore, to make these embeddings easier to visualize, researchers often apply dimensionality reduction techniques to project them into 2D or 3D space. Some popular dimensionality reduction techniques include UMAP (McInnes et al., 2020), t-SNE (van der Maaten and Hinton, 2008), and PCA (Pearson, 1901). Each of these techniques has its own strengths and weaknesses in terms of how well it preserves the embeddings’ global structure, its stochasticity, interpretability, and scalability. Despite these differences, all dimensionality reduction techniques produce data in the same structure. This

means users can choose any technique and visualize the projected embeddings with WIZMAP.

2.2 Interactive Embedding Visualization

Researchers have introduced interactive visualization tools to help users explore embeddings (e.g., Liu et al., 2018; Li et al., 2018; Arendt et al., 2020). For example, Embedding Projector (Smilkov et al., 2016) allows users to zoom, rotate, and pan 2D or 3D projected embeddings to explore and inspect data point features. Similarly, Deepscatter (Schmidt, 2021) and regl-scatterplot (Lekschas, 2023) empowers users to explore billion-scale 2D embeddings in their browsers. Latent Space Cartography (Liu et al., 2019) helps users find and refine meaningful semantic dimensions within the embedding space. In addition, researchers have designed visualizations to aid users in comparing embeddings, such as embComp (Heimerl et al., 2022) visualizing local and global similarities between two embeddings, Emblaze (Sivaraman et al., 2022) tracing the changes in the position of data points across two embeddings, and Embedding Comparator (Boggust et al., 2022) highlighting the neighborhoods around points that change the most across embeddings. In contrast, WIZMAP aims to help users navigate and interpret both the global and local structures of large embedding spaces by offering visual contexts at varying levels of granularity.

3 Multi-scale Embedding Summarization

Researchers have highlighted users’ desire for embedding visualizations to provide visual contexts and embedding summaries to facilitate exploration of various regions within the embedding space (Robertson et al., 2023). However, generating embedding summaries is challenging for two reasons. First, efficiently summarizing millions of data points in larger datasets can be a formidable task. Second, selecting the embedding regions to summarize is difficult, as users possess varying interests in regions of different sizes and levels of granularity. To tackle this challenge, we propose a novel method to automatically generate multi-resolution embedding summaries at scale.

Multi-resolution Quadtree Aggregation. First, we apply a dimensionality reduction technique such as UMAP to project high-dimensional embedding vectors into 2D points. From these points, we construct a quadtree (Finkel and Bentley, 1974), a tree data structure that recursively partitions a 2D space

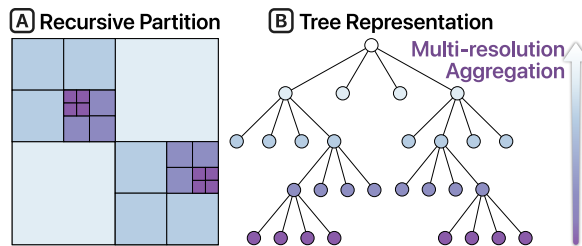


Fig. 3: (A) A quadtree recursively partitions a 2D space into four equally-sized squares, (B) and each square is represented as a tree node. WIZMAP efficiently aggregates information from the leaves to the root, summarizing embeddings at different levels of granularity.

into four equally-sized squares, each represented as a node. Each data point exists in a unique leaf node. To summarize embeddings across different levels of granularity, we traverse the tree bottom up. In each iteration, we first extract summaries of embeddings in each leaf node, and then merge the leaf nodes at the lowest level with their parent node. This process continues recursively, with larger and larger leaf nodes being formed until the entire tree is merged into a single node at the root. Finally, we map pre-computed embedding summaries to a suitable granularity level and dynamically show them as users zoom in or out in WIZMAP (§ 4.1).

Scalable Leaf-level Summarization. When performing quadtree aggregation, researchers have the flexibility to choose any suitable method for summarizing embedding from leaf nodes. For text embeddings, we propose t-TF-IDF (tile-based TF-IDF) that adapts TF-IDF (term frequency-inverse document frequency) to extract keywords from leaf nodes (Sparck Jones, 1972). Our approach is similar to c-TF-IDF (classed-based TF-IDF) that combines documents in a cluster into a meta-document before computing TF-IDF scores (Grootendorst, 2022). Here, we merge all documents in each leaf node (i.e., a tile in the quadtree partition) as a meta-document and compute TF-IDF scores across all leaf nodes. Finally, we extract keywords with the highest t-TF-IDF scores to summarize embeddings in a leaf node. This approach is scalable and complementary to quadtree aggregation. Because our document merging is hierarchical, we only construct the n-gram count matrix once and update it in each aggregation iteration with just one matrix multiplication. Summarizing 1.8 million text embeddings across three granularity levels takes only about 55 seconds on a MacBook Pro. For non-text data, we summarize embeddings by finding points closest to the embedding centroid in a leaf node.

4 User Interface

Leveraging pre-computed multi-resolution embedding summarization (§ 3), WIZMAP tightly integrates three interface components (Fig. 1A–C).

4.1 Map View

The *Map View* (Fig. 1A) is the primary view of WIZMAP. It provides a familiar map-like interface that allows users to pan and zoom to explore different embedding regions with varying sizes. To help users easily investigate both the global structure and local neighborhoods of their embeddings, the *Map View* integrates three layers of visualization.

Distribution Contour. To provide users with a quick overview of the global structure of their embeddings, we use Kernel Density Estimation (KDE) (Rosenblatt, 1956) to estimate the distribution of 2D embedding points. We use a standard multivariate Gaussian kernel with a Silverman bandwidth for the KDE model (Silverman, 2018). Next, we compute the distribution likelihoods over a 200×200 2D grid whose size is determined by the range of all embedding points. Finally, we visualize the likelihoods over the grid as a contour plot (Fig. 4), highlighting the high-level density distribution of users’ embeddings. Researchers can adjust the grid density, and we tune it by balancing the computation time and the contour resolution.

Multi-resolution Labels. The *Map View* helps users interpret embeddings across various levels of granularity by dynamically providing pre-computed contextual labels. It overlays summaries generated via quadtree aggregation (§ 3) onto the distribution contour and scatter plot. Users can hover over to see the summary from a quadtree tile closest to the cursor. Our tool adjusts the label’s tile size based on the user’s current zoom level. For example, when a user zooms into a small region, the *Map View* shows summaries computed at a lower level in the quadtree. In addition to on-demand embedding summaries, this view also automatically labels high-density regions (Fig. 4) by showing summaries from quadtree tiles near the geometric centers of high-probability contour polygons.

Scatter Plot. To help users pinpoint embeddings within their local neighborhoods, the *Map View* visualizes all embedding points in a scatter plot with their 2D positions.

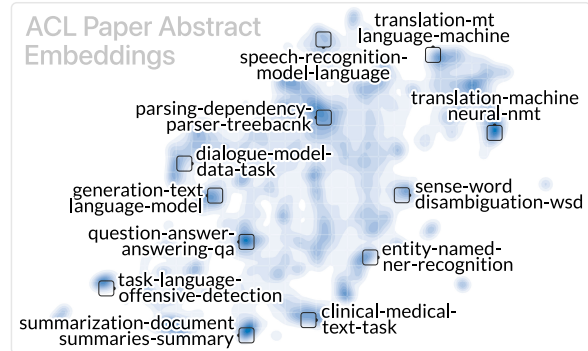
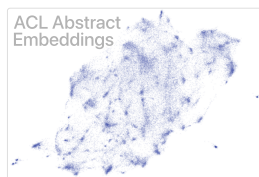
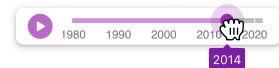


Fig. 4: The *Map View* provides an embedding overview via a contour plot and auto-generated multi-resolution embedding labels placed around high-density areas.

The corresponding scatter plot for Fig. 4 is shown on the right. Users can specify the color of each embedding point to encode additional features, such as the class of embeddings. Also, users can hover over an embedding point to reveal its original data, such as ACL paper abstracts (§ 5.1).

4.2 Control Panel

The *Map View* shows all three visualization layers



by default, and users can customize them to fit their needs by clicking buttons in the *Control Panel* (Fig. 1C). In addition, WIZMAP allows users to compare multiple embedding groups in the same embedding space by superimposing them in the *Map View* (Gleicher, 2018). In the case of embeddings that include times, users can use a slider (shown on the right) in the *Control Panel* to observe changes in the embeddings over time (Fig. 5).

4.3 Search Panel

Searching and filtering can help users discover interesting embedding patterns and test their hypothesis regarding the embedding structure (Carter et al., 2019). In WIZMAP, users can use the *Search Panel* (Fig. 1B) to search text embeddings including specified words in the original data. The panel shows a list of search results, and the *Map View* highlights their corresponding embedding points.

4.4 Scalable & Open-source Implementation

WIZMAP is scalable to *millions* of embedding points, providing a seamless user experience with zooming and animations, all within web browsers without backend servers. To achieve this, we leverage modern web technologies, especially WebGL to render embedding points with the `regl` API (Lysenko, 2016). We also use `Web Workers` and `Streams API` to enable the streaming of large em-

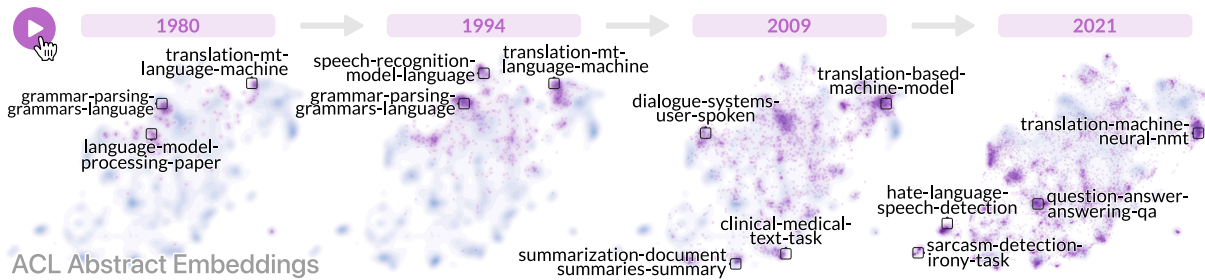



Fig. 5: WIZMAP allows users to observe how embeddings change over time. For example, when exploring 63k ACL paper abstracts, clicking the play button  in the *Control Panel* animates the visualizations to show embeddings of papers published in each year in purple and the distribution of all papers in blue. This animation highlights changes in ACL research topics over time, such as the decline in popularity of grammar and the rise of question-answering.

bedding files in parallel with rendering. To enable fast full-time search, we apply a contextual indexing algorithm with FlexSearch (Wilkerling, 2019). We use D3 (Bostock et al., 2011) for other visualizations and scikit-learn (Pedregosa et al., 2011) for KDE. To ensure that our tool can be easily incorporated into users’ current workflows, we apply NOVA (Wang et al., 2022b) to make WIZMAP available within computational notebooks. We provide detailed tutorials to help users use our tool with their embeddings. We have open-sourced our implementation to support future research and development of embedding exploration tools.

5 Usage Scenarios

We present two hypothetical scenarios, each with real embedding data, to demonstrate how WIZMAP can help ML researchers and domain experts easily explore embeddings and gain a better understanding of ML model behaviors and dataset patterns.


5.1 Exploring ACL Research Topic Trends

Helen, a science historian, is interested in exploring the evolution of computational linguistic and natural language processing (NLP) research since its inception. She downloads the Bibtex files of all papers indexed in ACL Anthology (Rohatgi, 2022), and extracts the paper title and abstract from 63k papers that have abstracts available. Then, Helen applies MPNet, a state-of-the-art embedding model (Song et al., 2020), to transform the concatenation of each paper’s title and abstract into a 768-dimensional embedding vector. She then trains a UMAP model to project extracted embeddings into a 2D space. She tunes the UMAP’s hyperparameter $n_neighbors$ to ensure projected points are spread out (Coenen and Pearce, 2019).

Helen uses a Python function provided by WIZMAP to generate three JSON files containing embedding summaries (§ 3), the KDE distribu-

tions (§ 4.1), and the original data in a streamable format (Hoeger et al., 2014). Helen configures the function to use the dataset’s year feature as the embedding’s time—the function computes the KDE distribution of embeddings for each year slice. She provides the files to WIZMAP and sees a visualization of all ACL abstract embeddings (Fig. 4A).

Embedding Exploration. In the *Map View*, Helen explores embeddings with zoom and pan. She also uses the *Search Panel* to find papers with specific keywords, such as “dialogue”, and Helen is pleased to see all related papers are grouped in a cluster (Fig. 1B). With the help of multi-resolution embedding summaries, Helen quickly gains an understanding of the structure of her embedding space. For example, she finds that the top right cluster features translation papers while the lower clusters feature summarization and medical NLP.

Embedding Evolution To examine how ACL research topics change over time, Helen clicks the play button clicking the play button  in the *Control Panel* to animate the visualizations. The *Map View* shows embeddings of papers published in each year from 1980 to 2022 in purple, while the distribution of all papers is shown as a blue background (Fig. 5). As Helen observes the animation, she identifies several interesting trends. For example, she observes a decline in the popularity of grammar research, while question-answering has become increasingly popular. She also notes the emergence of some small clusters in recent years, featuring relatively new topics, such as sarcasm, humor, and hate speech. Satisfied with the findings using WIZMAP, Helen decides to write an essay on the trend of NLP research over four decades.

5.2 Investigating Text-to-Image Model Usage

Bob, an ML researcher, works on improving text-to-image generative models. Recent advancements

in diffusion models, such as Stable Diffusion (Rombach et al., 2022), have attracted an increasing number of users to generate photorealistic images by writing text prompts. To gain an understanding of these models’ behaviors and identify potential weaknesses for improvement, Bob decides to study how users use these models in the wild by analyzing DiffusionDB, a dataset containing 14 million images generated by Stable Diffusion with 1.8 million unique text prompts (Wang et al., 2022a).

Bob’s analysis goal is to study the relationship between the text prompts and their generated images. Thus, he chooses to use CLIP (Radford et al., 2021) to encode both prompts and images into a 768-dimensional multimodal embedding before projecting them to a 2D space with UMAP. He uses prompts to generate embedding summaries for the CLIP space. After creating all JSON files, WIZMAP visualizes all 3.6 million embeddings (Fig. 6).

Embedding Exploration. Bob begins his exploration by hiding image embeddings and scatter plots, focusing on the global structure of embeddings with the contour plot and embedding summaries. He discovers two dominant prompt categories: art-related prompts and photography-related prompts. Two categories appear far from each other, which is not surprising as they are expected to have distinct semantic representations. Bob also notices two smaller clusters within the photography region, prompting him to zoom in and turn on the scatter plot to further investigate these local regions (Fig. 2). After hovering over a few points, he realizes one cluster is mostly about non-human objects while the other is about celebrities.

Embedding Comparison. To investigate the relationship between text prompts and their generated images, Bob clicks a button in the *Control Panel* to superimpose the contour and scatter plot of image embeddings in red onto the text embedding visualizations in blue (Fig. 6). Bob quickly identifies areas where two distributions overlap and differ. He notes that the “movie” cluster in the text embeddings has a lower density in the image embeddings, whereas a high-density “art portrait” cluster emerges in image embeddings. Bob hypothesizes that Stable Diffusion may have limited capability to generate photorealistic human faces (Borji, 2022). After exploring embedding with WIZMAP, Bob is pleased with his findings, and he will apply his insights to improve the curation of his training data.

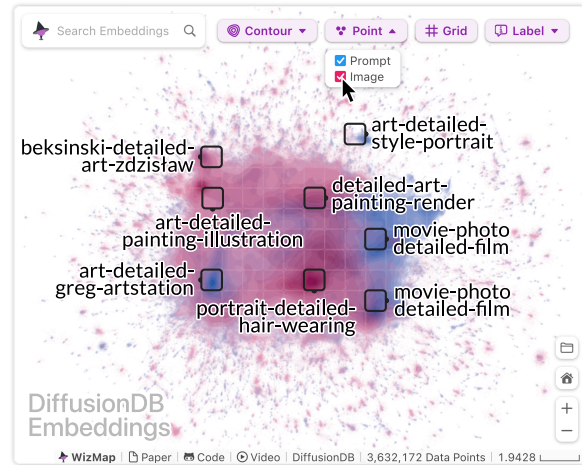


Fig. 6: WIZMAP enables users to compare multiple embeddings by visualization superposition. For instance, comparing the CLIP embeddings of **1.8 million** Stable Diffusion **prompts** and **1.8 million** generated **images** reveals key differences between two distributions.

6 Future Work and Conclusion

WIZMAP integrates a novel quadtree-based embedding summarization technique that enables users to easily explore and interpret large embeddings across different levels of granularity. Our usage scenarios showcase our tool’s potential for providing ML researchers and domain experts with a holistic view of their embeddings. Reflecting on our design and development of WIZMAP, we acknowledge its limitations and distill future research directions that could further assist users in interpreting and applying embeddings for downstream tasks.

- **User evaluation.** To investigate the usefulness of flexible transitioning across various levels of abstraction during embedding exploration, future researchers can use WIZMAP as a research instrument to conduct observational user studies with ML researchers and domain experts.
- **Automated insights.** Our tool provides automatic and multi-scale visual contexts to guide users in their exploration. While our quadtree-based approach is effective and scalable, it is sensitive to tile size selection. Researchers can explore more robust methods for embedding summarization and automated data insights, such as clustering-based approaches (Law et al., 2020).
- **Enhanced comparison.** WIZMAP helps users compare embedding groups through contour superposition. However, for local comparisons, other techniques such as juxtaposition and explicit encoding may be more effective (Gleicher, 2018). Future researchers can design visualization tools that integrate these techniques.

7 Broader Impact

We designed and develop WIZMAP with good intentions—to help ML researchers and domain experts easily explore and interpret large embeddings. However, bad actors could exploit insights gained from using WIZMAP for malevolent purposes. For example, research has shown that ML embeddings contain societal biases (Bolukbasi et al., 2016). Therefore, bad actors could manipulate and sabotage ML predictions by injecting inputs whose embeddings are known to associate with gender and racial biases. The potential harms of biased embeddings warrant further study.

Acknowledgements

We thank our anonymous reviewers for their insightful comments. This work was supported in part by a J.P. Morgan PhD Fellowship, Apple Scholars in AI/ML PhD fellowship, DARPA GARD, gifts from Cisco, Bosch, and NVIDIA. Use, duplication, or disclosure is subject to the restrictions as stated in Agreement number HR00112030001 between the Government and the Performer.

References

- Dustin L. Arendt, Nasheen Nur, Zhuanyi Huang, Gabriel Fair, and Wenwen Dou. 2020. [Parallel embeddings: A visualization technique for contrasting learned representations](#). In *Proceedings of the 25th International Conference on Intelligent User Interfaces*.
- Hedi Ben-younes, Remi Cadene, Nicolas Thome, and Matthieu Cord. 2019. [BLOCK: Bilinear Superdiagonal Fusion for Visual Question Answering and Visual Relationship Detection](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33.
- Angie Boggust, Brandon Carter, and Arvind Satyanarayan. 2022. [Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples](#). In *27th International Conference on Intelligent User Interfaces*.
- Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. [Man is to computer programmer as woman is to homemaker? Debiasing word embeddings](#). In *Advances in Neural Information Processing Systems*, volume 29.
- Ali Borji. 2022. [Generated Faces in the Wild: Quantitative Comparison of Stable Diffusion, Midjourney and DALL-E 2](#). *arXiv 2210.00586*.
- Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. [D³ Data-Driven Documents](#). *IEEE TVCG*, 17.
- Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. [Activation Atlas](#). *Distill*, 4.
- Andy Coenen and Adam Pearce. 2019. [Understanding UMAP](#).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *arXiv:1810.04805 [cs]*.
- Debidatta Dwivedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2021. [With a little help from my friends: Nearest-neighbor contrastive learning of visual representations](#). In *ICCV*.
- R. A. Finkel and J. L. Bentley. 1974. [Quad trees a data structure for retrieval on composite keys](#). *Acta Informatica*, 4.
- Michael Gleicher. 2018. [Considerations for Visualizing Comparison](#). *IEEE Transactions on Visualization and Computer Graphics*, 24.
- Maarten Grootendorst. 2022. [BERTopic: Neural topic modeling with a class-based TF-IDF procedure](#). *arXiv preprint arXiv:2203.05794*.
- Florian Heimerl, Christoph Kralj, Torsten Moller, and Michael Gleicher. 2022. [embComp : Visual Interactive Comparison of Vector Embeddings](#). *IEEE Transactions on Visualization and Computer Graphics*, 28.
- Thorsten Hoeger, Chris Dew, Finn Pauls, and Jim Wilson. 2014. [Newline Delimited JSON: A standard for delimiting JSON in stream protocols](#).
- Margaret L. Kern, Gregory Park, Johannes C. Eichstaedt, H. Andrew Schwartz, Maarten Sap, Laura K. Smith, and Lyle H. Ungar. 2016. [Gaining insights from social media language: Methodologies and challenges](#). *Psychological Methods*, 21.
- Po-Ming Law, Alex Endert, and John Stasko. 2020. [Characterizing Automated Data Insights](#). In *2020 IEEE Visualization Conference (VIS)*.
- Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. 2018. [CleanNet: Transfer learning for scalable image classifier training with label noise](#). In *CVPR*.
- Fritz Lekschas. 2023. [Regl-Scatterplot: A Scalable Interactive JavaScript-based Scatter Plot Library](#). *Journal of Open Source Software*, 8.
- Quan Li, Kristanto Sean Njotoprawiro, Hammad Haleem, Qiaoan Chen, Chris Yi, and Xiaojuan Ma. 2018. [EmbeddingVis: A Visual Analytics Approach to Comparative Network Embedding Inspection](#). *arXiv:1808.09074 [cs]*.
- Shusen Liu, Peer-Timo Bremer, Jayaraman J. Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. 2018. [Visual Exploration of Semantic Relationships in Neural Word Embeddings](#).

- IEEE Transactions on Visualization and Computer Graphics*, 24.
- Yang Liu, Eunice Jun, Qisheng Li, and Jeffrey Heer. 2019. [Latent Space Cartography: Visual Analysis of Vector Space Embeddings](#). *Computer Graphics Forum*, 38.
- Mikola Lysenko. 2016. [Regl: Functional WebGL](#).
- Leland McInnes, John Healy, and James Melville. 2020. [UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction](#). *arXiv:1802.03426 [cs, stat]*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient Estimation of Word Representations in Vector Space](#). *arXiv 1301.3781*.
- Haekyu Park, Nilaksh Das, Rahul Duggal, Austin P. Wright, Omar Shaikh, Fred Hohman, and Duen Horng Polo Chau. 2022. [NeuroCartography: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks](#). *IEEE Transactions on Visualization and Computer Graphics*, 28.
- Karl Pearson. 1901. [On lines and planes of closest fit to systems of points in space](#). *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. [Scikit-learn: Machine learning in python](#). *the Journal of machine Learning research*, 12.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *NAACL HLT*.
- 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](#). In *ICML*, volume 139 of *Proceedings of Machine Learning Research*.
- Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. 2019. [Transfusion: Understanding transfer learning for medical imaging](#). In *Advances in Neural Information Processing Systems*, volume 32.
- Samantha Robertson, Zijie J. Wang, Dominik Moritz, Mary Beth Kery, and Fred Hohman. 2023. [Angler: Helping Machine Translation Practitioners Prioritize Model Improvements](#). In *CHI Conference on Human Factors in Computing Systems*.
- Shaurya Rohatgi. 2022. [ACL anthology corpus with full text](#). Github.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. [High-resolution image synthesis with latent diffusion models](#). In *CVPR*.
- Murray Rosenblatt. 1956. [Remarks on Some Nonparametric Estimates of a Density Function](#). *The Annals of Mathematical Statistics*, 27.
- Benjamin Schmidt. 2021. [Deepscatter: Zoomable, animated scatterplots in the browser that scales over a billion points](#).
- Rita Sevastjanova, Eren Cakmak, Shauli Ravfogel, Ryan Cotterell, and Mennatallah El-Assady. 2022. [Visual Comparison of Language Model Adaptation](#). *IEEE Transactions on Visualization and Computer Graphics*.
- Bernard W Silverman. 2018. [Density Estimation for Statistics and Data Analysis](#).
- Venkatesh Sivaraman, Yiwei Wu, and Adam Perer. 2022. [Emblaze: Illuminating Machine Learning Representations through Interactive Comparison of Embedding Spaces](#). In *27th International Conference on Intelligent User Interfaces*.
- Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. 2016. [Embedding Projector: Interactive Visualization and Interpretation of Embeddings](#). *arXiv 1611.05469*.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tiejun Liu. 2020. [MpNet: Masked and permuted pre-training for language understanding](#). *Advances in Neural Information Processing Systems*, 33.
- Karen Sparck Jones. 1972. [A statistical interpretation of term specificity and its application in retrieval](#). *Journal of Documentation*, 28.
- Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. [Visualizing Large-scale and High-dimensional Data](#). In *Proceedings of the 25th International Conference on World Wide Web*.
- Laurens van der Maaten and Geoffrey Hinton. 2008. [Visualizing data using t-SNE](#). *Journal of Machine Learning Research*, 9.
- Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. 2022a. [DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models](#). *arXiv:2210.14896 [cs]*.
- Zijie J. Wang, David Munechika, Seongmin Lee, and Duen Horng Chau. 2022b. [NOVA: A Practical Method for Creating Notebook-Ready Visual Analytics](#). *arXiv:2205.03963 [cs]*.
- Thomas Wilkerling. 2019. [FlexSearch: Next-Generation full text search library for Browser and Node.js](#).

A System for Answering Simple Questions in Multiple Languages

Anton Razzhigaev^{1,2,*}, Mikhail Salnikov^{1,*}, Valentin Malykh³, Pavel Braslavski⁴,
and Alexander Panchenko^{1,2}

¹ Skolkovo Institute of Science and Technology ² Artificial Intelligence Research Institute

³ ISP RAS Research Center for Trusted Artificial Intelligence ⁴ Ural Federal University

Abstract

Our research focuses on the most prevalent type of queries—*simple questions*—exemplified by questions like “What is the capital of France?”. These questions reference an entity such as “France”, which is directly connected (one hop) to the answer entity “Paris” in the underlying knowledge graph (KG). We propose a multilingual Knowledge Graph Question Answering (KGQA) technique that orders potential responses based on the distance between the question’s text embeddings and the answer’s graph embeddings. A system incorporating this novel method is also described in our work.

Through comprehensive experimentation using various English and multilingual datasets and two KGs — Freebase and Wikidata — we illustrate the comparative advantage of the proposed method across diverse KG embeddings and languages. This edge is apparent even against robust baseline systems, including seq2seq QA models, search-based solutions and intricate rule-based pipelines. Interestingly, our research underscores that even advanced AI systems like ChatGPT encounter difficulties when tasked with answering simple questions. This finding emphasizes the relevance and effectiveness of our approach, which consistently outperforms such systems. We are making the source code and trained models from our study publicly accessible to promote further advancements in multilingual KGQA.

1 Introduction

A knowledge graph (KG) is a collection of *subject–predicate–object* triples, for example ⟨Paris, capital_of, France⟩. Large KGs are valuable resources for many tasks, including question answering (QA) (Ji et al., 2022). Knowledge graph question answering (KGQA) is an active research area, as well as a popular application.

Even though all major web search engines implement KGQA capabilities – KG results can be easily

recognized in their ‘smart answers’ – there are few operational KGQA research prototypes available online. A rare example is QAnswer (Diefenbach et al., 2020a), a rule-based KGQA system over Wikidata. There are also only a few free KGQA codebases available (Huang et al., 2019; Burtsev et al., 2018; Chen et al., 2021).

In this work, we focus on simple questions such as “What is the capital of France?”. There exists an opinion that the task of answering such questions is nearly solved (Petrochuk and Zettlemoyer, 2018), but openly available systems are scarce and do not support multiple languages. Besides, their performance, as will be observed from our work, is still far from perfect even for models based on deep neural networks specifically pre-trained on QA data. In our work, our aim is to address these limitations of the prior art.

We developed a KGQA method M3M (multilingual triple match) based on text-to-graph embedding search. The key idea illustrated in Figure 1 is to combine a pre-trained multilingual language model for question representation and pre-trained graph embeddings that represent KG nodes and edges as dense vectors. In the training phase, we learn separate projections of the question text embeddings to the subject, predicate, and object of the KG triple corresponding to the question-answer pair. In the test phase, we first fetch a set of candidate KG triples based on the question’s word n-grams and extract named entities to make the process more computationally efficient. Then, we rank candidate triples according to the sum of three cosine similarities – between the embeddings of the triple’s components and respective projections of the question’s embeddings. Finally, the object of the top-ranked triple is returned as an answer.

Our approach build upon Huang et al. (2019) expanding the method beyond a single KG and a single monolingual dataset. We experimented with the *de facto* standard English KGQA dataset

*The first two authors contributed equally.

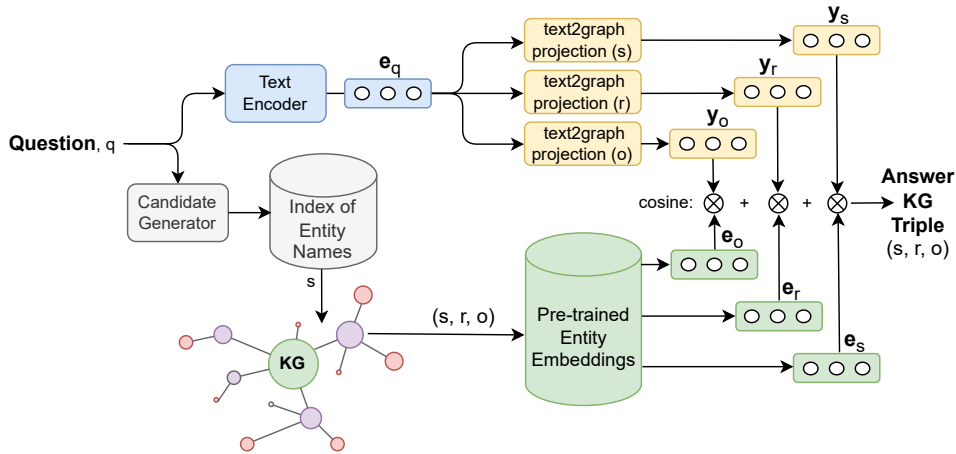


Figure 1: Workflow of M3M Knowledge Graph Question Answering system for simple questions.

Instance Of Entity	Count
Q484170 (commune of France)	17
Q1549591 (big city)	12
Q515 (city)	5
Q5119 (capital city)	5
Q1637706 (million city)	4
Q1187811 (college town)	4
Q51929311 (largest city)	3
Q486972 (human settlement)	2
Q22927616 (commune of France with specific status)	1

Figure 2: Graphical user interface of the KGQA system for answering one-hop questions.

SimpleQuestions, which is based on the now deprecated Freebase, to allow comparison with previous art. Moreover, we conducted experiments with several Wikidata-based datasets: SimpleQuestionsWd (a Wikidata mapping from the original benchmark), Russian/English RuBQ 2.0 dataset, as well as a recent Mintaka dataset covering nine languages. Our experiments demonstrate the applicability of the proposed method in different KGs and languages.

Our online demo (see Figure 2) implements two KGQA methods: (1) a T5 model fine-tuned on QA data and (2) our approach dubbed M3M based on embedding search. We believe that a combination of an online demo, publicly available code, as well as evaluation results on several datasets will contribute to future developments in the field of mul-

tilingual KGQA. To summarize, our contributions are the following:

- A novel multilingual approach to one-hop KGQA, which compares favorably to strong baselines, such as T5 QA system, and previous embedding-based methods on a battery of benchmarks.
- Open implementation of an online system for one-hop QA over Wikidata knowledge graph. We also release pre-trained models and provide an API making seamless integration into NLP applications possible.¹

¹Source code, link to the demo and video demonstration: <https://github.com/s-nlp/m3m>

2 Related Work

There are two main directions in KGQA research: semantic-parsing- and retrieval-based approaches. Semantic parsing seeks to map a natural language question to a meaning representation (e.g. SPARQL query language) that can be directly executed against a KG. Gu et al. (2022) provide a comprehensive survey of semantic parsing methods in KGQA. They also point out that KG schema richness (diversity of entity and relation types), irregularities and incompleteness of KG, as well as its large size make the application of semantic parsing methods to KGQA especially challenging.

The latter line of research follows information retrieval’s notion of proximity between the query and sought entity. Early methods relied on the lexical match of the question and the KG elements (entity labels, descriptions, and relation names) (Balog, 2018). In their pioneering work Bordes et al. (2014) proposed to embed question words and KG components in a shared space, with question representations close to the answers’ ones. This approach was further developed in a series of studies (Dai et al., 2016; Lukovnikov et al., 2017; Huang et al., 2019; Lukovnikov et al., 2019). Notably, more recent approaches leverage existing knowledge graph embeddings that have been used for a variety of KG tasks such as link prediction and entity classification (Wang et al., 2017) and pre-trained language models such as BERT applicable to a multitude of NLP tasks (Devlin et al., 2019). The embedding-based approach can be extended to complex KGQA (Saxena et al., 2020). A comprehensive overview of KGQA methods can be found in recent surveys (Chakraborty et al., 2021; Lan et al., 2021) and a monograph (Roy and Anand, 2021).

Large language models (LLMs) accumulate not only linguistic, but also factual knowledge, and thus can be considered as alternatives to KGs. Petroni et al. (2019) showed that BERT (a rather small model by today’s standards) possessed some relational knowledge without fine-tuning. Roberts et al. (2020) experimented with the T5 model (Raffel et al., 2020) as an open-domain QA system. In particular, after fine-tuning on question-answer pairs, at inference time, the model was given a question and returned an answer without accessing any external knowledge source. The results were quite encouraging; the model with “salient span masking” (SSM) pre-training proposed by Guu et al.

(2020) performed best. Mallen et al. (2022) found that LLMs memorise knowledge about common entities but fall short in the case of less common ones. They proposed to complement LLM-based QA with zero-shot and few-shot prompting with adaptive evidence retrieval. Tan et al. (2023) evaluate QA capabilities of FLAN-T5 (Chung et al., 2022) and several GPT versions (GPT3, GPT3.5, and chatGPT) in eight KGQA datasets. The results show that LLMs are yet quite far beyond SotA; chatGPT outperforms its contenders.

Multilingual KGQA has been relatively rarely addressed in the literature. Proposed solutions include question translation (Perevalov et al., 2022a), language-agnostic translation of syntactic parses into meaning representations (Reddy et al., 2017; Hakimov et al., 2017), and unsupervised induction of bilingual lexicons for word-level translation of training examples (Zhou et al., 2021). We hope that the advent of larger multilingual datasets with Wikidata annotations (Longpre et al., 2021; Sen et al., 2022) will stimulate interest in this timely topic.

3 M3M: Multilingual Triple Match QA

In this section, we describe the core of our contribution: a novel method for one-hop question answering over a knowledge graph. The proposed method dubbed M3M for Multilingual Triple Match relies on matching independently KGs subjects, objects, and predicates with the questions’ representation. Overall, the approach can be considered as an improved version of the method by Huang et al. (2019) with which we perform extensive comparisons in the remainder of the paper among other baselines.

3.1 Question Answering Method

The architecture of the M3M model is illustrated in Figure 1. M3M uses multilingual BERT (Devlin et al., 2019) as question encoder. mBERT is a case-sensitive “base” 12-layer Transformer model that was trained with a masked language model objective on a collection of 104 Wikipedias. We encode the question q with 768-dimensional mBERT-based embeddings (M_{enc}) averaging the last hidden states of all tokens $e_q = M_{enc}(q)$. To project these representations to graph embeddings, we use three 2-layered perceptrons with ELU as an activation function. These three perceptrons are used to predict the embeddings of the object, relation, and

subject of the answer triple. We train four models simultaneously with AdamW optimiser with default parameters and $lr = 1e - 4$ for 10 epochs. We use MSE loss between the predicted and ground truth graph embeddings.

At the test phase we first generate a set of candidate triples. We extract named entities from the question using an off-the-shelf NER tool, as well as all word uni-, bi-, and trigrams if there are no named entities found. We use SpaCy² as the base entity detection model and lemmatizer. After extracting all candidate entity mentions from the question, we search the collection of KG labels and retain all entities containing at least one of the extracted question parts as a substring in their labels. On average, the list contains ~ 300 candidates.

Then, we map the question embedding e_q to the KG embedding space with three models (M_s, M_r, M_o), thus obtaining three projected embeddings of the question: e_s, e_r, e_o . We calculate the corresponding cosine similarities between the projected embeddings and KG embeddings of the candidate triples. Then, we sum these three cosine similarities, obtaining the answer score of each candidate triple. As graph embeddings are normalised and the model was trained with MSE loss, a sum of cosine similarities is an appropriate choice for a search metric. The QA system returns a ranked list of triples with the corresponding scores. Finally, the object of the highest-ranked triple is considered the answer. The overall schema of the proposed method for one-hop question answering is summarised in Algorithm 1.

3.2 System Implementation

The demo web application illustrated in Figure 2 is implemented using FastAPI framework.³ It contains a Web application backend that can be used with any KGQA system over Wikidata. The application’s frontend communicates with an asynchronous Web API (see Swagger documentation in Figure 7 in Appendix). The end user communicates with each of the provided pipelines by sending an HTTP POST request to `/pipelines/model` with a JSON object containing the question in the “text” field. Every pipeline responds to the end user with a JSON object containing an “answers” field with a sorted array of Wikidata identifiers. In addition to the “answers”, the pipeline responses

can include additional information, for example, `/pipelines/m3m` returns “scores” and “uncertainty”. The entire KGQA system can be launched on any platform that supports Docker.⁴ This demo can be easily integrated into other applications using the API endpoints provided.

We employ a conventional user interface for our Web demo – the KGQA system returns a ranked list of results along with links to Wikidata. In addition, the system response provides a visualization of the intermediate stages and sub-graphs corresponding to the questions, which helps to better understand the results. The core M3M method is based on a deeply reworked implementation of QA method firstly mentioned in (Chekalina et al., 2022) featuring various optimizations related to (1) effective candidate selection, and (2) retrieval of candidates and triples.

Algorithm 1 M3M method for one-hop KGQA

Input: q – question, \mathcal{G} – KG, \mathbf{E} – KG embeddings, M_{enc} – text encoder, M_s, M_r, M_o – projection modules, NER – subject candidates extractor
Output: $\langle o_a, r_a, s_a \rangle$ – triple with subject answer entity s_a

```

 $e_q = M_{enc}(q)$ 
 $\mathbf{A}=[]$  ▷ Initialize answer-candidates
 $\mathbf{S}=[]$  ▷ Initialize scores
 $\mathbf{C}=[]$  ▷ Initialize entity-candidates
candidates = NER( $q$ )
for entity in  $\mathcal{G}$  do
  if entity.name in candidates then  $\mathbf{C}.append(\text{entity})$ 
for entity in  $\mathbf{C}$  do
  for relation in entity.relations do
     $s = \text{entity.id}; r = \text{relation.id}; o = \text{entity}[r]$ 
     $\mathbf{A}.append(\langle s, r, o \rangle)$ 
     $\mathbf{e}_s = \mathbf{E}[s]; \mathbf{e}_r = \mathbf{E}[r]; \mathbf{e}_o = \mathbf{E}[o]$ 
     $\mathbf{y}_s = M_s(\mathbf{e}_q); \mathbf{y}_r = M_r(\mathbf{e}_q); \mathbf{y}_o = M_o(\mathbf{e}_q)$ 
     $score = \cos(\mathbf{e}_o, \mathbf{y}_o) + \cos(\mathbf{e}_r, \mathbf{y}_r) + \cos(\mathbf{e}_s, \mathbf{y}_s)$ 
     $\mathbf{S}.append(score)$ 
ind = argmax( $\mathbf{S}$ )
 $\langle s_a, r_a, o_a \rangle = \mathbf{A}[\text{ind}]$ 
return  $\langle s_a, r_a, o_a \rangle$ 

```

4 Experiments

In this section, we present the benchmarking of the presented system on several one-hop KGQA datasets for Freebase and Wikidata knowledge graphs. The method is compared to strong baselines. Both the datasets and the baselines are described below.

4.1 Data

Knowledge graphs. Freebase knowledge graph (Bollacker et al., 2008), launched in 2007, was a pioneering project that was developed to a large extent by community members. In 2010

²

³<https://fastapi.tiangolo.com>

⁴<https://www.docker.com>

Freebase was acquired by Google, and in 2015 it was discontinued as an independent project. The ‘frozen’ knowledge base dump was made publicly available and was also incorporated into Wikidata (Tanon et al., 2016). Nowadays Wikidata (Vrandečić and Krötzsch, 2014) is the largest and most up-to-date open KG. At the time of writing, Wikidata contains more than 100 million unique entities, 7,500 relation types, and 12.6 billion statements.⁵ Wikidata items are provided with labels and descriptions in multiple languages. Many KGQA datasets have been using Freebase as an underlying knowledge graph. However, we can observe an increasing uptake of Wikidata as a basis for KGQA research. Wikidata underlies recent English (Cao et al., 2022; Dubey et al., 2019), as well as several multilingual datasets (Longpre et al., 2021; Rybin et al., 2021; Sen et al., 2022; Perevalov et al., 2022b). In our experiments, we use Freebase2M (FB2M) (Bordes et al., 2015) and Wikidata8M (Korablinov and Braslavski, 2020)⁶ KG snapshots containing 2M and 8M entities, respectively.

KG embeddings. In our experiments, we use TransE KG embeddings that represent both entities and relations as vectors in the same space (Bordes et al., 2013). For KG triples $\langle s, p, o \rangle$ corresponding embeddings are expected to allow the following ‘translation’: $s + p \approx o$. We conducted experiments on two TransE models: (1) TransE embeddings with $\text{dim}=250$ based on Freebase2M snapshot provided by Huang et al. (2019); and (2) TransE embeddings with $\text{dim}=200$ trained on the full Wikidata using Pytorch-BigGraph framework (Lerer et al., 2019).

QA datasets. In contrast to open-domain question answering and machine reading comprehension, there are rather few studies devoted to the multilingual KGQA (Perevalov et al., 2022a). This can partly be explained by the lack of data: until recently, KGQA datasets were almost entirely English, and rare exceptions such as QALD (Perevalov et al., 2022b) contained only hundreds of questions, which is rather scarce for training modern data-hungry models. Larger-scale datasets with KG annotations such as MKQA (Longpre et al., 2021) and Mintaka (Sen et al., 2022) have appeared only recently.

⁵<https://www.wikidata.org/wiki/Property:P10209>

⁶<https://zenodo.org/record/3751761>

To be able to compare our method with its predecessors, we use SimpleQuestions (Bordes et al., 2015). The dataset consists of 108,442 questions formulated by English speakers using Freebase triples as prompts. The question usually mentions the triple’s subject and predicate, while the object is the expected answer. Note that by design the dataset contains many ambiguous questions having multiple answers in the KG, while only one answer is considered correct, see analysis in (Petrochuk and Zettlemoyer, 2018; Wu et al., 2020). Further, we make use of SimpleQuestionsWd, a mapping of SimpleQuestions to Wikidata (Diefenbach et al., 2017). The dataset contains 16,414/4,751 questions in train/test sets, respectively. We filtered the dataset and retained only questions, whose triples are present in Wikidata8M which resulted in 8,327 train and 2,438 test questions. RuBQ (Rybin et al., 2021) is a KGQA data set that contains 2,910 Russian questions of different types along with their English translations. Due to its limited size, the official RuBQ split has only development and test subsets. For our experiments, we kept only 1-hop questions that constitute the majority of the dataset. Mintaka (Sen et al., 2022) is a recently published dataset that contains 20k questions of different types translated into eight languages. For our experiments, we took only *generic* questions, whose entities are one hop away from the answers’ entities in Wikidata, which resulted in 1,236/181/340 train/dev/test questions in each language, respectively.⁷ Table 1 summarizes the data we used for training and evaluation.

Dataset	KG	Lang.	Train	Test
SimpleQuestions (SQ)	FB	en	75,910	21,687
SimpleQuestionsWd	WD	en	8,327	2,438
RuBQ	WD	ru, en	300	1,186
Mintaka-Simple	WD	mult	1,236	340

Table 1: Datasets in our experiments (FB – Freebase, WD – Wikidata).

4.2 Baselines

We compare our system with three groups of methodologically diverse state-of-the-art QA systems: based on rule-based approach, relying on

⁷These questions are not necessarily *simple* in terms of SimpleQuestions dataset. E.g. both entities (*Heath Ledger*, *The Dark Knight*) in the question *Who did Heath Ledger play as in the movie The Dark Knight?* are one hop away from the answer entity *Joker*; both triples are essential to provide a correct answer.

embedding search similar to our approach, and generative neural models, e.g. sequence-to-sequence.

QAnswer is a rule-based multilingual QA system proposed by [Diefenbach et al. \(2020b\)](#). It returns a ranked list of Wikidata identifiers as answers and a corresponding SPARQL query. We use QAnswer API in our experiments.⁸

KEQA: Knowledge Embedding based Question Answering. There are no published results on SimpleQuestions aligned with Wikidata KG which is why we adopt the official implementation of KEQA ([Huang et al., 2019](#)) – an open-source embedding-based KGQA solution to SimpleQuestionsWd benchmark. It was initially trained and evaluated on Freebase embeddings. To the best of our knowledge, there are no open-sourced KGQA models with better performance than KEQA. We use this model as the main baseline on SimpleQuestionsWd and original SimpleQuestions benchmarks. To make a comparison with KEQA on the SimpleQuestionsWd test set more fair, we re-train it on PTBG-Wikidata embeddings. We use the official implementation with provided hyperparameters and an internal validation mechanism.⁹ As SimpleQuestionsWd is a subset of the original SimpleQuestions, we evaluate Freebase-pretrained KEQA on this dataset as well (taking into account the corresponding entity mapping).

T5-based QA system. Question answering can also be addressed as a seq2seq task. To provide a comparison with this type of approaches, we conducted experiments with T5, an encoder-decoder transformer-based model pre-trained on a multi-task mixture of unsupervised and supervised tasks ([Raffel et al., 2020](#)). T5 works well on a variety of tasks out-of-the-box by prepending a prefix to the input corresponding to each of the tasks. To answer English questions, we used T5 model fine-tuned on a large *NaturalQuestions* dataset ([Roberts et al., 2020](#)). For other languages, we fine-tuned mT5-xl model ([Xue et al., 2021](#)) on Mintaka Simple.

In addition, we carried out experiments employing the Flan-T5-xl ([Chung et al., 2022](#)) model, a recent development trained on a diverse mixture of tasks. We evaluated this model in

⁸<https://qanswer-frontend.univ-st-etienne.fr>

⁹https://github.com/xhuang31/KEQA_WSDM19

two distinct setups using the Mintaka dataset: firstly in a zero-shot setting, utilizing the prompt “Question: question Answer:”, and secondly, by separately fine-tuning the model on the training data for each individual language.

GPT-3 has gained recognition for its impressive performance in both few-shot and zero-shot contexts ([Brown et al., 2020](#)), excelling in a vast array of benchmarks. A recent study ([Chung et al., 2022](#)) evaluate different GPT versions on complex KG questions. However, the experiments don’t include datasets in our study. To address this oversight and offer a comparative baseline for our system, we subjected the GPT-3 model (davinci-003) to the SimpleQuestionsWd and RuBQ 2.0 benchmarks. Detailed information on the generation parameters and prompts can be found in the Appendix.

ChatGPT stands as one of the leading systems in the field of Natural Language Processing (NLP), demonstrating capabilities for intricate reasoning and extensive factual knowledge ([OpenAI, 2023](#)). We evaluated this system (GPT-3.5-turbo-0301) using the RuBQ 2.0 and SimpleQuestionsWd benchmarks. Specifics about the prompts and generation parameters are available in the Appendix.

4.3 Experimental Setup

To compare our algorithm with baselines, we use the Accuracy@1 metric i.e. correctness of the first retrieved result. The answer of a QA system to an answerable question is considered correct if its object matches the answer in terms of Wikidata id or just by a label string.

It is essential to acknowledge that sequence-to-sequence (seq2seq) models yield a string instead of a knowledge graph ID, which may pose a challenge during evaluation. To mitigate this, we apply specific transformations to the responses produced by seq2seq systems. These include converting the text to lowercase and eliminating any leading and trailing spaces. This transformation process is also applied to label-aliases representing the actual answers present in the RuBQ and Mintaka datasets.

Regarding the SimpleQuestionsWd dataset, we procure aliases for the correct answers via the Wikidata API.¹⁰ We then determine the accuracy of the seq2seq model’s prediction by checking for an exact match between the predicted string and one of the aliases.

¹⁰<https://pypi.org/project/Wikidata>

Model	SQ	SQ-WD	RuBQ-ru	RuBQ-en
QAnswer (Diefenbach et al., 2020a)	–	33.31	30.80	32.30
T5-11b-ssm-nq, fine-tuned (Roberts et al., 2020)	–	20.40	–	42.75
ChatGPT – GPT-3.5-turbo	–	17.75	26.99	30.12
GPT-3 – davinci-003	–	28.51	18.10	34.20
KEQA (Huang et al., 2019) – TransE FB2M	75.40	40.48	–	–
KEQA (Huang et al., 2019) – TransE PTBG	–	48.89	–	33.80
M3M (Ours) – TransE FB2M	76.90 \pm 0.30	–	–	–
M3M (Ours) – TransE PTBG	–	53.50 \pm 0.30	48.40 \pm 0.30	49.50 \pm 0.30

Table 2: Comparison of M3M system with KGQA baselines in terms of Accuracy@1 for monolingual one-hop QA datasets. The best scores are highlighted. M3M scores are averages over models trained with five random seeds.

Model	en	es	de	ar	fr	pt	it	hi	avg
mT5-xl, fine-tuned (Xue et al., 2021)	20.8	19.5	19.3	12.6	19.7	18.3	20.9	9.7	17.6
FlanT5-xl, fine-tuned (Chung et al., 2022)	35.3	22.0	23.3	0.2	25.0	24.0	25.5	0	19.41
FlanT5-xl, zero-shot (Chung et al., 2022)	14.7	6.5	7.6	0	0.7	0.9	0.9	0	3.19
M3M (Ours) – TransE PTBG	26.0	26.1	25.0	24.1	25.0	24.7	25.3	24.1	25.0

Table 3: Results on Mintaka-Simple dataset (one-hop questions) for models trained simultaneously on all languages.

It is noteworthy to mention that in the Mintaka-Simple test set, about a half of the answers don’t have labels in Hindi.

4.4 Results

Table 2 contains the results of our M3M model and several baselines on two versions of the Simple-Questions dataset and two versions of the RuBQ dataset. Specifically, for the RuBQ dataset, we detail the outcomes derived from testing both Russian and English language queries.

Interestingly, ChatGPT, despite being recognized as a more sophisticated system, exhibits a weaker performance on factoid questions compared to GPT-3. Upon conducting a concise manual error analysis, we observed that ChatGPT frequently dismissed queries with responses such as “Answer is unknown”, or sought supplemental information. We suggest that this behavior may be a consequence of the system’s alignment with human feedback, implemented to limit the model’s tendency for generating ungrounded or ‘hallucinated’ responses. However, it is plausible that a more refined prompt design could address this issue and enhance the system’s performance on such questions. Nonetheless, this exploration extends beyond the scope of our current research and is suggested as an avenue for further investigation.

Table 3 features the results obtained on the Mintaka-Simple dataset, providing an opportunity to evaluate the mT5, Flan-T5 and M3M models. This table highlights the multilingual capabili-

ties exhibited by both the generative and the KG-retrieval approaches. An analysis of these results reveals that our model’s performance is markedly stable across languages, indicating a lesser dependence on the language relative to the seq2seq approach. Our model manifests exceptional performance on one-hop simple questions and achieves a new state-of-the-art on the RuBQ 2.0 (Russian) benchmark as well as on the English SimpleQuestionsWd dataset. These findings illustrate the superiority of KG-based models, outperforming both GPT-3 and ChatGPT by a considerable margin.

5 Conclusion

In this study, we introduced M3M, a multilingual model, along with an open implementation, devised for one-hop knowledge base question answering. Our approach leverages the use of a multilingual text encoder and pre-trained KG embeddings, which are aligned using a triple projection method of a question to subject/relation/object of KG triple to facilitate efficient answer search in the embedding space.

For simple questions, our system not only outperforms previous strong alternatives, including rule-based approaches, embeddings-based similarity search, and pre-trained sequence-to-sequence neural models, but also excels when compared to advanced AI models like ChatGPT. These comparative results were drawn from a comprehensive battery of one-hop QA datasets, including both monolingual and multilingual data.

6 Limitations

While a large fraction of users' information needs may be fulfilled by answering simple questions, the main limitation of the proposed system is that in the current implementation, it can be applied only to one-hop KG questions. As it may be not obvious to figure out beforehand which question is one-hop and which is multi-hop in a KG special classifiers or uncertainty estimation techniques should be ideally combined with the proposed system to not let the system answer questions it is not designed to answer in the first place. At the same time, our preliminary experiments with training classifiers of question type based on Mintaka data show promising results, suggesting that such classifier effectively could be created and used in real deployments in conjunction with the proposed system.

In terms of computational efficiency, communication with a knowledge graph can be a bottleneck if based on a public SPARQL endpoint with query limits but could be substantially sped up using an in-house SPARQL engine or using indexing of triples with appropriate data structures. However, in the latter case, a mechanism for updating such structures is required to keep system answers up to date.

7 Ethical Statement

QA systems built on top of large pretrained neural models, such as those described in this paper, may transitively reflect biases available in the training data potentially generating stereotyped answers to questions. It is therefore recommended in production (as compared to research settings) to use a special version of debiased pre-trained neural models and/or deploy a special layer of debiasing systems around the proposed methodology.

Acknowledgements

Pavel Braslavski's work was supported in part by the Ministry of Science and Higher Education of the Russian Federation (project 075-02-2023-935). The work of Valentin Malykh was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142.

References

- Krisztian Balog. 2018. *Entity-Oriented Search*, volume 39 of *The Information Retrieval Series*. Springer.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. *Freebase: A collaboratively created graph database for structuring human knowledge*. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, page 1247–1250, New York, NY, USA. Association for Computing Machinery.
- Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. *Question answering with subgraph embeddings*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620. ACL.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. *Large-scale simple question answering with memory networks*. *CoRR*, abs/1506.02075.
- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. *Translating embeddings for modeling multi-relational data*. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.
- 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.
- Mikhail S. Burtsev, Alexander V. Seliverstov, Rafael Airapetyan, Mikhail Y. Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lyman, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. *Deeppavlov: Open-source library for dialogue systems*. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 122–127. Association for Computational Linguistics.

- Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. [KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.
- Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. 2021. [Introduction to neural network-based question answering over knowledge graphs](#). *WIREs Data Mining Knowl. Discov.*, 11(3).
- Viktoriia Chekalina, Anton Razzhigaev, Albert Sayapin, Evgeny Frolov, and Alexander Panchenko. 2022. [MEKER: Memory efficient knowledge embedding representation for link prediction and question answering](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 355–365, Dublin, Ireland. Association for Computational Linguistics.
- Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. [ReTraCk: A flexible and efficient framework for knowledge base question answering](#). In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 325–336. Association for Computational Linguistics.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. [Scaling instruction-finetuned language models](#). *CoRR*, abs/2210.11416.
- Zihang Dai, Lei Li, and Wei Xu. 2016. [CFO: Conditional focused neural question answering with large-scale knowledge bases](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 800–810, Berlin, Germany. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). 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.
- Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. 2020a. [Towards a question answering system over the semantic web](#). *Semantic Web*, 11(3):421–439.
- Dennis Diefenbach, José M. Giménez-García, Andreas Both, Kamal Singh, and Pierre Maret. 2020b. [Qanswer KG: designing a portable question answering system over RDF data](#). In *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 429–445. Springer.
- Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. 2017. [Question answering benchmarks for wikidata](#). In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, volume 1963 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. [Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia](#). In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 69–78. Springer.
- Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. [Knowledge base question answering: A semantic parsing perspective](#). *CoRR*, abs/2209.04994.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. [Retrieval augmented language model pre-training](#). In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.
- Sherzod Hakimov, Soufian Jebbara, and Philipp Cimiano. 2017. [AMUSE: multilingual semantic parsing for question answering over linked data](#). In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 329–346. Springer.
- Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. [Knowledge graph embedding based question answering](#). In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 105–113. ACM.
- Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. [A survey on knowl-](#)

- edge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2):494–514.
- Vladislav Korablinov and Pavel Braslavski. 2020. **RuBQ: A Russian dataset for question answering over Wikidata**. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 97–110. Springer.
- Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. **A survey on complex knowledge base question answering: Methods, challenges and solutions**. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4483–4491. ijcai.org.
- Adam Lerer, Ledell Wu, Jiajun Shen, Timothée Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. **Pytorch-BigGraph: A large scale graph embedding system**. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org.
- Shayne Longpre, Yi Lu, and Joachim Daiber. 2021. **MKQA: A linguistically diverse benchmark for multilingual open domain question answering**. *Trans. Assoc. Comput. Linguistics*, 9:1389–1406.
- Denis Lukovnikov, Asja Fischer, and Jens Lehmann. 2019. **Pretrained transformers for simple question answering over knowledge graphs**. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, pages 470–486.
- Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. **Neural network-based question answering over knowledge graphs on word and character level**. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1211–1220. ACM.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khoshdel. 2022. **When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories**. *CoRR*, abs/2212.10511.
- OpenAI. 2023. **GPT-4 technical report**. *CoRR*, abs/2303.08774.
- Aleksandr Perevalov, Andreas Both, Dennis Diefenbach, and Axel-Cyrille Ngonga Ngomo. 2022a. **Can machine translation be a reasonable alternative for multilingual question answering systems over knowledge graphs?** In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 977–986. ACM.
- Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022b. **Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers**. In *16th IEEE International Conference on Semantic Computing, ICSC 2022, Laguna Hills, CA, USA, January 26-28, 2022*, pages 229–234. IEEE.
- Michael Petrochuk and Luke Zettlemoyer. 2018. **SimpleQuestions nearly solved: A new upperbound and baseline approach**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 554–558, Brussels, Belgium. Association for Computational Linguistics.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. **Language models as knowledge bases?** In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. **Universal semantic parsing**. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.
- Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. **How much knowledge can you pack into the parameters of a language model?** In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.
- Rishiraj Saha Roy and Avishek Anand. 2021. **Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections**. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.
- Ivan Rybin, Vladislav Korablinov, Pavel Efimov, and Pavel Braslavski. 2021. **RuBQ 2.0: An innovated russian question answering dataset**. In *The Semantic Web - 18th International Conference, ESWC 2021*, pages 532–547.
- Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. **Improving multi-hop question answering over knowledge graphs using knowledge base embeddings**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

- Priyanka Sen, Alham Fikri Aji, and Amir Saffari. 2022. [Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1604–1619, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. [Evaluation of chatgpt as a question answering system for answering complex questions](#). *CoRR*, abs/2303.07992.
- Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. [From freebase to wikidata: The great migration](#). In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1419–1428. ACM.
- Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: a free collaborative knowledgebase](#). *Commun. ACM*, 57(10):78–85.
- Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. [Knowledge graph embedding: A survey of approaches and applications](#). *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.
- Zhiyong Wu, Ben Kao, Tien-Hsuan Wu, Pengcheng Yin, and Qun Liu. 2020. [PERQ: predicting, explaining, and rectifying failed questions in KB-QA systems](#). In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, pages 663–671. ACM.
- 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.
- Yucheng Zhou, Xiubo Geng, Tao Shen, Wenqiang Zhang, and Daxin Jiang. 2021. [Improving zero-shot cross-lingual transfer for multilingual question answering over knowledge graph](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5822–5834, Online. Association for Computational Linguistics.

A Parameter Settings and Prompts for OpenAI Models

The parameters for GPT-3 (davinci-003) and ChatGPT (GPT-3.5-turbo-0301) were configured to a temperature setting of 0.1, while the top_p for GPT-3 was set to 0.85. The prompts for both models, which are illustrated in Figure 3, were used respectively: for GPT-3 directly, and for ChatGPT, conveyed via the “user” field in the chat API interface.

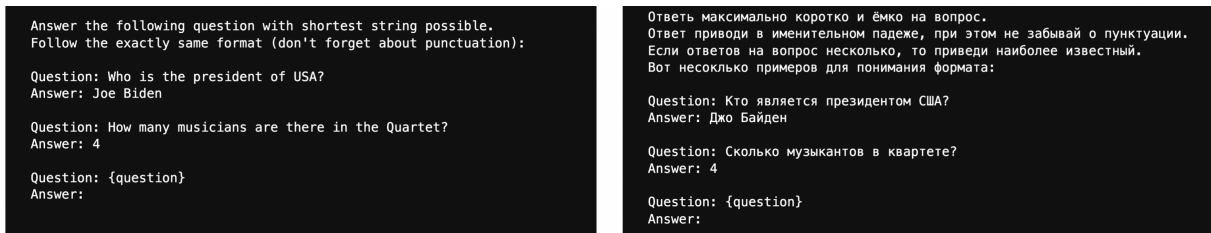


Figure 3: Prompts employed for ChatGPT and GPT-3 in the English and Russian language benchmarks, respectively.

B Additional Illustrations of the Knowledge Graph Question Answering System

This section contains three illustrations of the graphical user interface with three additional questions as well as a demonstration of the API for integration into external applications.

The image shows the KGQA system interface. At the top, there is a search bar with the query "What is capital of Germany?" and an "Ask" button. Below the search bar, the system displays results for the question. On the left, there is a card for "Berlin" with a description and a list of related entities. On the right, there is a knowledge graph showing the relationship between "Germany (Q183)" and "Berlin (Q64)" with the label "capital". Below the graph, there is a table of "Instance Of Entity" and a list of related entities.

Search Results:

- <https://www.wikidata.org/wiki/Q64>
Berlin
federal state, capital and largest city of Germany
Q133442 (city-state) Q200250 (metropolis) Q707813 (Hanseatic city) Q1221156 (federated state of Germany) Q1549591 (big city)
- <https://www.wikidata.org/wiki/Q1055>
Hamburg
city and federal state in the North of Germany
Q1221156 (federated state of Germany) Q1549591 (big city) Q1637706 (million city) Q2264924 (port settlement)

Knowledge Graph:

```

graph TD
    G[Germany (Q183)] -- capital --> B[Berlin (Q64)]
    B -- country --> G
    
```

Question entities: Q183 (Germany)

Final Instance Of: Q1549591 (big city) Q42744322 (urban municipality of Germany)

Instance Of Entity	Count
Q1549591 (big city)	16
Q42744322 (urban municipality of Germany)	16
Q14784328 (state capital in Germany)	6
Q253030 (major regional center)	6
Q1187811 (college town)	5
Q200250 (metropolis)	4
Q707813 (Hanseatic city)	4
Q1637706 (million city)	4
Q515 (city)	4

Figure 4: Graphical user interface of KGQA system for the one-hop question “What is capital of Germany?”.

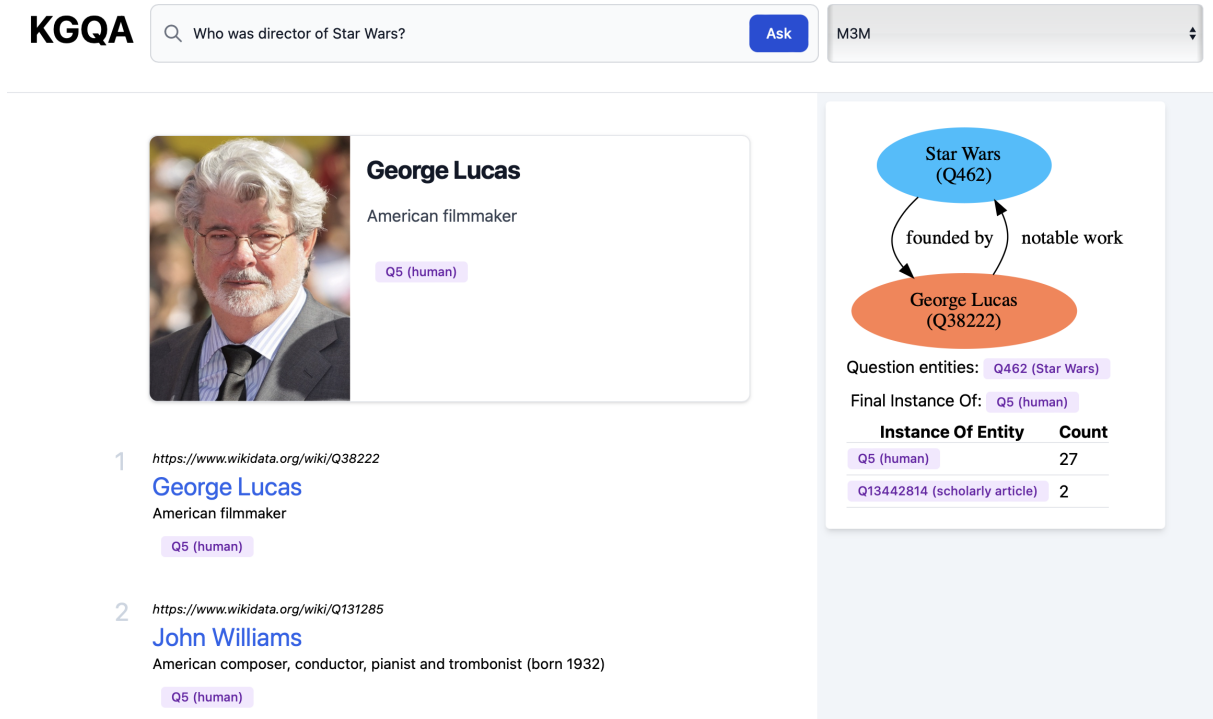


Figure 5: Graphical user interface of KGQA system for the one-hop question “Who was director of Star Wars?”.

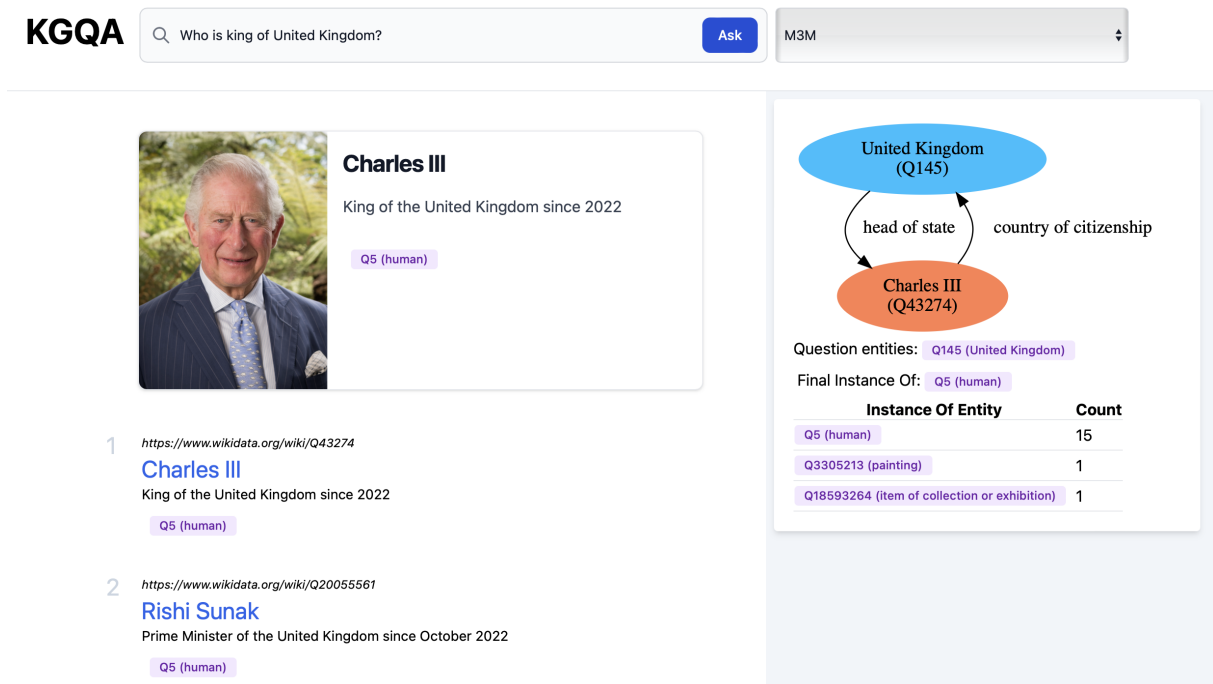


Figure 6: Graphical user interface of KGQA system for the one-hop question “Who is the king of United Kingdom?”.

pipeline		^
POST	/pipeline/act_selection/ner	Ner To Sentence Insertation
POST	/pipeline/act_selection/mgenre	Mgenre Linking
POST	/pipeline/act_selection/entity_selection	Entity Selection Mgenre Postprocess
POST	/pipeline/act_selection/seq2seq	Raw Seq2Seq
POST	/pipeline/act_selection/main	Pipeline
POST	/pipeline/act_selection/simple_type_selection/	Pipeline
POST	/pipeline/seq2seq/	Seq2Seq Pipeline
POST	/pipeline/m3m/	M3M Pipeline

Figure 7: Swagger API for the developed system allowing for integration of the KGQA functionality into applications (e.g. “seq2seq” or “m3m” endpoints) as well as subcomponents, such as NER for questions or type selection.

The screenshot displays the Swagger API interface for the `POST /pipeline/m3m/` endpoint. The interface includes a header with the endpoint name and a 'Try it out' button. Below the header, there is a 'Parameters' section indicating 'No parameters'. The 'Request body' section is marked as 'required' and has a dropdown menu set to 'application/json'. An 'Example Value' section shows a JSON object: `{ "text": "string" }`. The 'Responses' section features a table with one entry: a 200 status code for a 'Successful Response'. Below this, there is a 'Media type' dropdown set to 'application/json', a note 'Controls Accept header.', and another 'Example Value' section showing a JSON response: `{ "answers": ["string"], "scores": [0], "uncertainty": 0 }`.

Figure 8: Swagger API of an individual endpoint: parameters for KGQA method are documented and can be called using a RESTful endpoint.

KWJA: A Unified Japanese Analyzer Based on Foundation Models

Nobuhiro Ueda¹, Kazumasa Omura¹, Takashi Kodama¹,
Hirokazu Kiyomaru¹, Yugo Murawaki¹, Daisuke Kawahara², Sadao Kurohashi¹

¹ Kyoto University, Kyoto, Japan,

² Waseda University, Tokyo, Japan

{ueda, omura, kodama, kiyomaru, murawaki, kuro}@nlp.ist.i.kyoto-u.ac.jp,
dkw@waseda.jp

Abstract

We present KWJA, a high-performance unified Japanese text analyzer based on foundation models. KWJA supports a wide range of tasks, including typo correction, word segmentation, word normalization, morphological analysis, named entity recognition, linguistic feature tagging, dependency parsing, PAS analysis, bridging reference resolution, coreference resolution, and discourse relation analysis, making it the most versatile among existing Japanese text analyzers. KWJA solves these tasks in a multi-task manner but still achieves competitive or better performance compared to existing analyzers specialized for each task. KWJA is publicly available under the MIT license at <https://github.com/ku-nlp/kwja>.

1 Introduction

End-to-end neural network-based models have become mainstream for many NLP tasks including machine translation (Sutskever et al., 2014; Luong et al., 2015; Vaswani et al., 2017) and dialogue response generation (Serban et al., 2015; Roller et al., 2020). However, end-to-end models are not always the best means of developing an NLP application because exploratory tasks, such as information analysis and causal analysis, inherently require manual trial-and-error processes. We believe that for such tasks, text analysis still plays an important role.

Text analysis, including morphological analysis, dependency parsing, predicate-argument structure (PAS) analysis, and discourse relation analysis, saw shifts in model architectures. Recent studies demonstrate that foundation models (Bommasani et al., 2021) drastically improve the performance in dependency parsing (Zhou and Zhao, 2019), PAS analysis (Ueda et al., 2020; Umakoshi et al., 2021), and discourse relation analysis (Kishimoto et al., 2020; Kiyomaru and Kurohashi, 2021). Moreover, improvements on foundation models tend to have a greater impact on performance than incremental

improvements tailored to individual tasks (Bommasani et al., 2021).

In this study, we design and build a unified Japanese text analyzer, KWJA,^{1,2} in view of the fact that recent high-performance text analysis models are all based on foundation models. KWJA supports a wide variety of text analysis tasks: typo correction, word segmentation, word normalization, morphological analysis, named entity recognition, linguistic feature tagging, dependency parsing, PAS analysis, bridging reference resolution, coreference resolution, and discourse relation analysis (Figure 1, Table 1).

Our emphasis is on usability in addition to performance. KWJA provides a single command to perform a variety of text analyses, collapsing the painstaking steps previously needed to obtain the same result, namely, installing and combining multiple text analyzers, one for each task.

The design policy of KWJA is to minimize the amount of code and hand-written rules by maximally exploiting the power of foundation models. This is a drastic departure from the traditional Japanese analysis suite, including the morphological analyzers JUMAN (Kurohashi et al., 1994)³ and Juman++ (Tolmachev et al., 2018)⁴ and the dependency parser KNP (Kurohashi and Nagao, 1994),⁵ which rely heavily on manually constructed dictionaries, rules, and features. Such lexical knowledge is context insensitive and suffers from limited coverage. Motivated by the observation that foundation models learn massive knowledge through pre-training on large raw corpora, we narrow our efforts to supervised learning from relatively small annotated corpora. This approach enables us to support a new task just by constructing an annotated

¹Kyoto-Waseda Japanese Analyzer.

²Video demo: https://youtu.be/p2x_IrSmS20

³<https://github.com/ku-nlp/juman>

⁴<https://github.com/ku-nlp/jumanpp>

⁵<https://github.com/ku-nlp/knp>

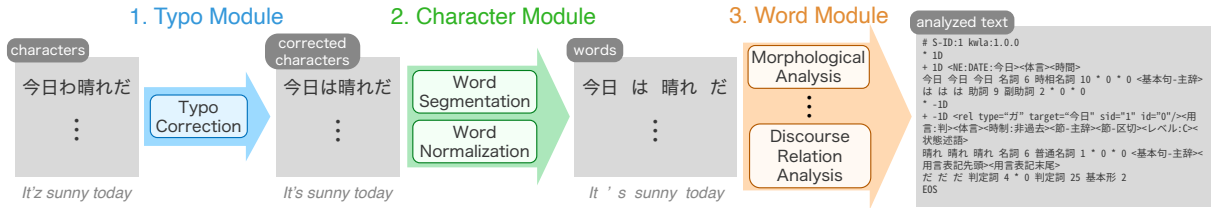


Figure 1: A flowchart of the analysis process in KWJA. KWJA consists of the typo module, character module, and word module. An input text is processed through each module.

Analysis Component	Input	Output Unit	Output
Typo Correction	characters	character	corrected characters
Word Segmentation	corrected characters	word	words
Word Normalization		word	normalized words
Morphological Analysis	words	word	POS, conjugation, lemma, and reading
Named Entity Recognition		word	named entity spans and categories
Linguistic Feature Tagging		word	word features
Dependency Parsing		base phrase	base phrases features
		word	dependency tree and dependency types
PAS Analysis		base phrase	dependency tree and dependency types
Bridging Reference Resolution		base phrase	predicates and their arguments
Coreference Resolution		base phrase	anaphors and their referents
Discourse Relation Analysis		base phrase	coreferring mentions
		clause	discourse relations

Table 1: Input and output of each analysis component. A base phrase is a unit consisting of a single independent word and its ancillary words (Hangyo et al., 2012). We group the components into three blocks (separated by horizontal lines) for multi-task learning.

corpus for the task.

KWJA reorganizes text analysis tasks into a dozen of components, as shown in Table 1, on the ground of task independence and the units of inputs and outputs. A notable difference from conventional morphological analysis is that while it is usually defined as the joint task of word segmentation, part-of-speech (POS) tagging, and the tagging of lexical information, such as lemmas and readings, we divide the task into word segmentation and the remaining tasks. For convenience, we refer to the latter as morphological analysis.

Each analysis component utilizes a fine-tuned Transformer. As Transformer consumes a considerable amount of computational resources, we resort to multi-task learning to reduce the model parameters. We group the components into three modules: the typo module, character module, and word module. Within each module, analysis components share most of their model parameters and run concurrently. Consequently, KWJA executes all the components in three steps (Figure 1).

Although KWJA is extremely slow for users who only need word segmentation, it is the most

practical choice for advanced text analysis, for which, after all, only Transformer achieves state-of-the-art performance. KWJA is publicly available under the MIT license at <https://github.com/ku-nlp/kwja>.

2 Related Work

Traditionally, Japanese text analysis tasks have been tackled individually, with separate analyzers for each task. Juman++ (Tolmachev et al., 2018) and Mecab (Kudo et al., 2004) are examples of morphological analyzers, while KNP (Kurohashi and Nagao, 1994) is a dependency parser. Juman++ and Mecab segment Japanese text into words and assign lexical information to each word using manually constructed dictionaries. KNP assigns dependency relations between phrases using linguistic features obtained from Juman++ and external dictionaries. In addition to dependency parsing, KNP handles named entity recognition, linguistic feature tagging, and PAS analysis. KWJA, however, supports an even broader range of tasks.

The Universal Dependencies (UD) project (Nivre et al., 2020) standardizes the

annotation of dependency structures across languages. While their focus is on dependency relations, the UD guidelines define word units, POS tags, and other linguistic features. The tasks supported by major UD-based analyzers, UD-Pipe (Straka et al., 2016), spaCy,⁶ and Stanza (Qi et al., 2020), are sentence segmentation, word segmentation, POS tagging, lemmatization, and dependency parsing.⁷ In other words, higher-level tasks such as PAS analysis and discourse relation analysis are out of the scope of these analyzers. A major advantage of UD-based analyzers is that they can handle multiple languages. This is done at the expense of ignoring language-specific features (Kanayama et al., 2018). For the purpose of pioneering task design, it is reasonable to focus on a single language.

GiNZA⁸ is also a UD-based analyzer but specializes in Japanese. GiNZA supports morphological analysis, dependency parsing, and named entity recognition. GiNZA v5 improved the dependency parsing performance by utilizing the foundation model ELECTRA.

Kachako (Kano, 2013) and Jigg (Noji and Miyao, 2016) have been proposed as frameworks for combining existing analysis tools to form a pipeline. These works aim to improve the usability of existing analysis tools. In contrast, our goal is to design and build a unified analyzer itself.

3 Resources

This section presents the model and data resources used when training the modules in KWJA.

3.1 Foundation Models

As a foundation model, we adopt DeBERTa (He et al., 2021), which has shown high performance in the SuperGLUE language understanding benchmark (Wang et al., 2019). We pre-trained character-level⁹ and word-level¹⁰ DeBERTa V2 large models on Japanese texts. The typo and character module employs the character-level model, and the word module employs the word-level model.

⁶<https://spacy.io>

⁷Using extra resources, spaCy and Stanza support named entity recognition in some languages.

⁸<https://github.com/megagonlabs/ginza>

⁹<https://huggingface.co/ku-nlp/deberta-v2-large-japanese-char-wwm>

¹⁰<https://huggingface.co/ku-nlp/deberta-v2-large-japanese>

3.2 Annotated Corpora

We use the Japanese Wikipedia Typo Dataset (JWTD v2, Tanaka et al., 2021) to train a typo correction model. JWTD was created by mining typos from the edit history of Japanese Wikipedia.

We use the Kyoto University Text Corpus (KC, Kurohashi and Nagao, 1998),¹¹ the Kyoto University Web Document Leads Corpus (KWDL, Hangyo et al., 2012),¹² and the Annotated Fuman Kaitori Center Corpus (Fuman)¹³ to train models for tasks other than typo correction. Note that as for discourse relation analysis, we use only KWDL because the other two corpora do not have discourse relation annotations.

4 Architecture

Each analysis component of KWJA uses a Transformer-based (Vaswani et al., 2017) foundation model. We add two layers of feed-forward neural networks for each task and fine-tune the whole model.

KWJA formulates the text analysis tasks as a sequence labeling task, word selection task, or word relation classification task. A sequence labeling task assigns a label to each character or word in a text. Figure 2 shows an example of solving named entity recognition as a sequence labeling task. In a word selection task, a word is selected from a text for each given word in the text. Figure 3 shows an example of solving dependency parsing as a word selection task. A word relation classification task assigns a label to each word pair in a text.

To reduce computational time and space, the model parameters of the analyzer are shared as much as possible through multi-task learning. The tasks in each block separated by the horizontal lines in Table 1 are the unit of multi-task learning. Multi-task learning is not possible for the pair of word segmentation and morphological analysis, for example, because the latter’s input depends on the former’s output. In this study, we perform multi-task learning for tasks in each block. Thus, KWJA consists of three modules corresponding to each block. These modules are referred to as the typo, character, and word modules, respectively.

While morphological analysis and dependency parsing use a sentence as the smallest unit of analysis, PAS analysis and discourse relation analysis

¹¹<https://github.com/ku-nlp/KyotoCorpus>

¹²<https://github.com/ku-nlp/KWDL>

¹³<https://github.com/ku-nlp/AnnotatedFKCCorpus>

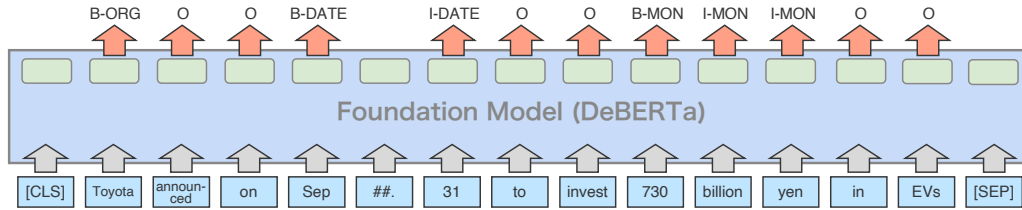


Figure 2: An example of solving named entity recognition as a sequence labeling task.

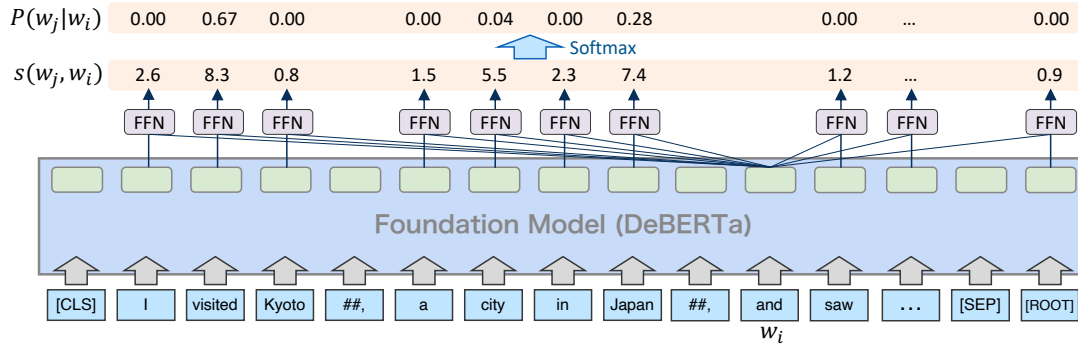


Figure 3: An example of solving dependency parsing as a word selection task.

require document-level processing. In this study, we apply document-level processing for all tasks to perform multi-task learning. With this formulation, obtaining sentence boundaries becomes less important. Thus, we only use simple rules based on regular expressions for sentence segmentation.

5 Analysis Components

KWJA consists of eleven analysis components belonging to three modules. In this section, we describe the details of each component, including its definition, formulation, and evaluation.

5.1 Typo Module

5.1.1 Typo Correction

Typo correction is the task of detecting and correcting typos in a text. Tanaka et al. (2021) used a pre-trained seq2seq model to convert input text to typo-corrected text. While seq2seq models enable flexible typo correction, they are at risk of grossly deviating from the input text.

To reduce the risk, we take a conservative approach; we formulate the task as a sequence labeling task where two edit operations (Malmi et al., 2019) are assigned to each character. Specifically, for each character in an input text, the model (1) chooses one from {KEEP, DELETE, REPLACE: x } and (2) predicts INSERT: x . REPLACE: x is an operation of replacing the character with x . INSERT: x inserts x before the character (x can be null). Follow-

ing Tanaka et al. (2021), we use the F1 score of character-level minimum edits for evaluation.

5.2 Character Module

5.2.1 Word Segmentation

Word segmentation is the task of splitting a text into words. We formulate the task as a character-level sequence labeling task. This task assigns a B (Begin) or I (Inside) label to each character in a text. The evaluation metric is the F1 score for word spans.

5.2.2 Word Normalization

Word normalization is the task of normalizing non-standard notations such as “Thank youuuuu” in place of “Thank you.” As with typo correction, we formulate the task as a sequence labeling task where a normalization operation is assigned to each character. The list of normalization operations is shown in Appendix A. The evaluation metric is the micro-averaged F1 score over labels other than KEEP, given that KEEP overwhelms the others.

5.3 Word Module

5.3.1 Morphological Analysis

In this study, we refer to morphological analysis as the task of assigning a POS, a sub-POS, a conjugation type, a conjugation form, a lemma, and a reading to each word. We formulate the first four tasks as word-level sequence labeling tasks. A post-

processing code is used to generate the lemma of a word by looking up its normalized surface form, conjugation type, and conjugation form in the conjugation table. We also formulate the task of assigning readings as a sequence labeling task, where the label set is the subword vocabulary defined in a pre-trained model. This is somewhat complicated because we have to preprocess the training data to split a reading into two or more if the corresponding word is split into multiple subwords. To do this, we perform alignment of character sequences of words and readings in accordance with subwords.

5.3.2 Named Entity Recognition

Named entity recognition is the task of identifying named entities in a text. We formulate the task as a word-level sequence labeling task. Following Akbik et al. (2018), we add a CRF layer on top of a foundation model. Named entities have the eight categories defined in the IREX CRL named entity data (Sekine and Isahara, 2000). The evaluation metric is the micro-averaged F1 score over the named entity categories.

5.3.3 Linguistic Feature Tagging

Linguistic feature tagging is the task of assigning various linguistic features to each word or base phrase.¹⁴ We formulate the task as a word-level sequence labeling task. Base phrase linguistic features are assigned to the head word of each base phrase. The evaluation metric is the macro-averaged F1 score over the features.

As the existing corpora do not have manually annotated linguistic features, we assign silver features using a rule-based Japanese linguistic feature tagger, KNP (Kurohashi and Nagao, 1994). In the future, we will manually correct some of the features and use them as gold data. All the features we used are listed in Appendix B.

5.3.4 Dependency Parsing

In this study, dependency parsing consists of two sub-tasks; one recognizes syntactic dependencies between words, and the other identifies their dependency types. We formulate the former task as a word selection task, following Zhang et al. (2017), and the latter task as a word relation classification task. As the evaluation metric, we use the Labeled Attachment Score (LAS) for base phrases.

¹⁴A unit consisting of a single independent word and its ancillary words. One or more base phrases make up a phrase.

5.3.5 PAS Analysis, Bridging Reference Resolution, and Coreference Resolution

PAS analysis, bridging reference resolution, and coreference resolution are the tasks of recognizing semantic relations between base phrases. PAS analysis finds arguments corresponding to *who* did/does *what* to *whom* for a predicate. Bridging reference resolution finds nouns that complement the essential information of another noun. Coreference resolution finds a set of nouns that refer to the same real-world entity.

Following Ueda et al. (2020), we formulate all the tasks as a word selection task. In PAS analysis, we focus on four cases: nominative, accusative, dative, and nominative-2.¹⁵

5.3.6 Discourse Relation Analysis

Discourse relation analysis is the task of recognizing discourse relations between clauses. Following Kawahara et al. (2014), we assign a label to each clause pair in a text. Note that clauses are identified with linguistic feature tagging.

We target the following discourse relations: CAUSE/REASON, PURPOSE, CONDITION, JUSTIFICATION, CONTRAST, and CONCESSION. In addition, we introduce a special relation NORELATION, which indicates none of the above relations is applicable, and formulate the task as a seven-way word relation classification task. We use the micro-averaged F1 score of the labels other than NORELATION as the evaluation metric.

6 Experiments and Discussion

In this section, we investigate the performance of each analysis component through fine-tuning foundation models.

6.1 Experimental Settings

We trained models on all the training data of KC, KWDL, and Fuman, and evaluated the performance per corpus. The details of the experimental settings are described in Appendix C.

6.2 Results

The result of each task is shown in Table 2. Overall, the performance of KWJA was comparable to SOTA and was sufficient for practical use. However, the F1 score of word normalization was 33.3, which was remarkably lower than those of the other

¹⁵Nominative-2 is used for a common Japanese construction in which a predicate has two nominative arguments.

Task	Corpus	Metric	Reference	SOTA	KWJA	
Typo Correction	JWTD	F1	Tanaka et al. (2021)	77.6	83.1±0.3	
Word Segmentation	KC	F1	Tolmachev et al. (2020)	99.5	99.3±0.1	
Word Normalization	all	F1	—	—	33.3±0.0	
Morphological Analysis	POS	KC	F1	Tolmachev et al. (2020)	99.1	99.7±0.1
	sub-POS	KC	F1	Tolmachev et al. (2020)	97.8	99.0±0.1
	conjugation type	all	F1	—	—	99.3±0.3
	conjugation form	all	F1	—	—	99.5±0.2
	reading	all	Accuracy	—	—	95.8±0.7
Named Entity Recognition	all	F1	—	—	84.3±4.0	
Linguistic Feature Tagging	word	all	F1	—	—	98.6±0.1
	base phrase	all	F1	—	—	88.3±3.1
Dependency Parsing	KC	LAS	Kawahara and Kurohashi (2006)	90.4	92.7±0.4	
PAS Analysis	all	F1	Ueda et al. (2020)	77.4	75.9±1.5	
Bridging Reference Resolution	all	F1	Ueda et al. (2020)	64.3	65.8±1.6	
Coreference Resolution	all	F1	Ueda et al. (2020)	67.4	77.7±0.9	
Discourse Relation Analysis	KWDLC	F1	Omura and Kurohashi (2022)	51.9	41.7±0.9	

Table 2: The performance of KWJA on each task in comparison to the state-of-the-art (SOTA). We fine-tuned KWJA with 3 different random seeds and report the mean and standard deviation of the performance. “—” indicates that no previous studies reported the performance on the corpora we used. “all” indicates KC, KWDLC, and Fuman corpus, and the metric is the macro-average of them.

tasks. Moreover, PAS analysis and discourse relation analysis scores were more than 1 point lower than SOTA.

6.3 Discussion

We discuss word normalization, PAS analysis, and discourse relation analysis in the following sections. Section 6.3.3 compares the analysis speed of KWJA with existing Japanese analyzers.

6.3.1 Word Normalization

The F1 score of word normalization, 33.3, was strikingly low. We attribute the poor performance to the highly unbalanced label distribution. Word normalization mainly targets informal texts, and there were very few examples with labels other than KEEP in the annotated corpora. We generated pseudo training data by applying denormalization rules to randomly selected words. Even with the pseudo-data, the percentage of labels other than KEEP was less than 0.1%, however. We plan to expand training data by specifically targeting low-frequency phenomena.

Analyzer	Time
Juman++ & KNP	1.1min
Juman++ & KNP (w/ PAS analysis)	18.4min
KWJA (ours)	2.7min

Table 3: Time spent by KWJA to analyze 1k sentences, with comparison to Juman++ (Tolmachev et al., 2018) and KNP (Kurohashi and Nagao, 1994).

6.3.2 PAS Analysis and Discourse Relation Analysis

We hypothesized that the low performance of PAS analysis and discourse relation analysis was due to multi-task learning, in which the model’s capability was allocated to the other tasks. To test this hypothesis, we trained the model separately for each task using single-task learning. The F1 score of PAS analysis was 79.3 ± 1.0 , and that of discourse relation analysis was 55.3 ± 3.6 . Both scores were significantly higher, confirming the hypothesis. We plan to adjust the loss weights and provide an option to use models trained with single-task learning. Appendix D shows the results of single-task learning for all tasks in the word module.

6.3.3 Speed of Analysis

We compared KWJA with the existing Japanese analyzers, Juman++ (Tolmachev et al., 2018) and KNP (Kurohashi and Nagao, 1994), in terms of speed of analysis. For KWJA, we performed all the tasks it supports. Juman++ supports word segmentation and morphological analysis while KNP supports named entity recognition, linguistic feature tagging, dependency parsing, and optionally, PAS analysis. We used 1k sentences randomly sampled from the Japanese portion of the CC-100 corpus (Wenzek et al., 2020). We used an NVIDIA TITAN V 12GB GPU to run KWJA.

Table 3 shows the results. We can see that KWJA was considerably faster than Juman++ and KNP even though KWJA performed a larger number of tasks.

7 Conclusion

In this study, we designed and built a unified Japanese text analyzer, KWJA, on top of foundation models. KWJA supports typo correction, word segmentation, word normalization, morphological analysis, named entity recognition, linguistic feature tagging, dependency parsing, PAS analysis, bridging reference resolution, coreference resolution, and discourse relation analysis in a unified framework. Users can quickly obtain analysis results by inputting a text and specifying the desired level of analysis.

One of the advantages of KWJA is its simplified design, thanks to the use of foundation models. Various analysis tasks, previously solved separately, are now performed only with three modules. For further simplification, we plan to solve all the analysis tasks with a character-level foundation model.

Limitations

As KWJA is based on a large Transformer model, the analysis in an environment without GPUs is expected to be slow. Even in environments with GPUs, when we need only a specific task (e.g., word segmentation), existing analyzers might be faster with little difference in accuracy.

The experiments showed that multi-task learning decreased accuracy in PAS analysis and discourse relation analysis. This fact may be true for other tasks as well. Therefore, when very high analysis accuracy is required for a particular task, using a model trained only on that task is recommended instead of KWJA.

Acknowledgements

The pre-training of foundation models used in this work was supported by the Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN) through General Collaboration Project no. jh221004 and jh231006, “Developing a Platform for Constructing and Sharing of Large-Scale Japanese Language Models.” As for the computing environment, we used the mdx: a platform for the data-driven future.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. [Contextual String Embeddings for Sequence Labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avani Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogun, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. [On the Opportunities and Risks of Foundation Models](#).
- Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. [Building a Diverse Document](#)

- Leads Corpus Annotated with Semantic Relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali, Indonesia. Faculty of Computer Science, Universitas Indonesia.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. **DeBERTa: Decoding-Enhanced BERT with Disentangled Attention**. In *International Conference on Learning Representations*.
- Hiroshi Kanayama, Na-Rae Han, Masayuki Asahara, Jena D. Hwang, Yusuke Miyao, Jinho D. Choi, and Yuji Matsumoto. 2018. **Coordinate Structures in Universal Dependencies for Head-final Languages**. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 75–84, Brussels, Belgium. Association for Computational Linguistics.
- Yoshinobu Kano. 2013. **Kachako: A Hybrid-Cloud Unstructured Information Platform for Full Automation of Service Composition, Scalable Deployment and Evaluation - Natural Language Processing as an Example**. In *Service-Oriented Computing - ICSOC 2012 Workshops - ICSOC 2012, International Workshops ASC, DISA, PAASC, SCEB, SeMaPS, WESOA, and Satellite Events, Shanghai, China, November 12-15, 2012, Revised Selected Papers*, volume 7759 of *Lecture Notes in Computer Science*, pages 72–84. Springer.
- Daisuke Kawahara and Sadao Kurohashi. 2006. **A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis**. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, New York City, USA. Association for Computational Linguistics.
- Daisuke Kawahara, Yuichiro Machida, Tomohide Shibata, Sadao Kurohashi, Hayato Kobayashi, and Manabu Sassano. 2014. **Rapid Development of a Corpus with Discourse Annotations using Two-stage Crowdsourcing**. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 269–278, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Yudai Kishimoto, Yugo Murawaki, and Sadao Kurohashi. 2020. **Adapting BERT to Implicit Discourse Relation Classification with a Focus on Discourse Connectives**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1152–1158, Marseille, France. European Language Resources Association.
- Hirokazu Kiyomaru and Sadao Kurohashi. 2021. **Contextualized and Generalized Sentence Representations by Contrastive Self-Supervised Learning: A Case Study on Discourse Relation Analysis**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5578–5584, Online. Association for Computational Linguistics.
- Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. **Applying conditional random fields to Japanese morphological analysis**. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.
- Sadao Kurohashi and Makoto Nagao. 1994. **KN Parser: Japanese Dependency/Case Structure Analyzer**. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 48–55.
- Sadao Kurohashi and Makoto Nagao. 1998. **Building a Japanese Parsed Corpus while Improving the Parsing System**. In *Proceedings of the NLPRS*, pages 719–724.
- Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. **Improvements of Japanese Morphological Analyzer JUMAN**. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 22–38.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. **Effective Approaches to Attention-based Neural Machine Translation**. *ArXiv*, abs/1508.04025.
- Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. **Encode, Tag, Realize: High-Precision Text Editing**. 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 5054–5065, Hong Kong, China. Association for Computational Linguistics.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. **Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.
- Hiroshi Noji and Yusuke Miyao. 2016. **Jigg: A Framework for an Easy Natural Language Processing Pipeline**. In *Proceedings of ACL-2016 System Demonstrations*, pages 103–108, Berlin, Germany. Association for Computational Linguistics.
- Kazumasa Omura and Sadao Kurohashi. 2022. **Improving Commonsense Contingent Reasoning by Pseudo-data and Its Application to the Related Tasks**. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 812–823, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. **Stanza: A Python Natural Language Processing Toolkit for Many Human Languages**. In *Proceedings of the 58th Annual*

- Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.
- Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y.-Lan Boureau, and Jason Weston. 2020. Recipes for Building an Open-Domain Chatbot. In *Conference of the European Chapter of the Association for Computational Linguistics*.
- Satoshi Sekine and Hitoshi Isahara. 2000. **IREX: IR & IE Evaluation Project in Japanese**. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).
- Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *AAAI Conference on Artificial Intelligence*.
- Milan Straka, Jan Hajič, and Jana Straková. 2016. **UD-Pipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing**. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. *Advances in neural information processing systems*, 27.
- Yu Tanaka, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. 2021. **Building a Japanese Typo Dataset and Typo Correction System Based on Wikipedia's Revision History**. *Journal of Natural Language Processing*, 28(4):995–1033. (in Japanese).
- Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. **Juman++: A Morphological Analysis Toolkit for Scriptio Continua**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium. Association for Computational Linguistics.
- Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2020. **Design and Structure of The Juman++ Morphological Analyzer Toolkit**. *Journal of Natural Language Processing*, 27(1):89–132.
- Nobuhiro Ueda, Daisuke Kawahara, and Sadao Kurohashi. 2020. **BERT-based Cohesion Analysis of Japanese Texts**. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1323–1333, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Masato Umakoshi, Yugo Murawaki, and Sadao Kurohashi. 2021. **Japanese Zero Anaphora Resolution Can Benefit from Parallel Texts Through Neural Transfer Learning**. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1920–1934, Punta Cana, Dominican Republic. 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.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. **SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems**. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- 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 Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.
- 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.
- Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. **Dependency Parsing as Head Selection**. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.
- Junru Zhou and Hai Zhao. 2019. **Head-Driven Phrase Structure Grammar Parsing on Penn Treebank**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

A Word Normalization Operations

We define six types of normalization operations as follows:

KEEP Keep the original character.

DELETE Delete the character.

VOICED Replace voiced or semi-voiced character with voiceless character (e.g., “*か*” (*ga*) → “*カ*” (*ka*), “*ぱ*” (*pa*) → “*ハ*” (*ha*)). This reverts *rendaku*, the voicing of the initial consonant of a non-initial word of a compound.

SMALL Replace a small character with the large character (e.g., “*な**あ*” → “*な**ア*”).

PROLONG Replace prolonged sound mark with its equivalent hiragana or katakana (e.g., “*も**れ**つ*” (*moRrets**u*) → “*も**う**れ**つ*” (*mourets**u*)).

PROLONG-E Replace a prolonged sound mark with “*え*” (*e*) (e.g., “*ね**ー*” (*neR*) → “*ねえ*” (*nee*)).

B Word and Base Phrase Features

KWJA assigns the following linguistic features.^{16,17} † indicates that the feature is to be corrected manually in the future.

Word Features

- base phrase head†
- base phrase end†, phrase end†
- declinable head or end

Base Phrase Features

- verbal (verb, adjective, copula)†, nominal†
- stative predicate, active predicate
- nominal predicate (verb, adjective)†
- modality†, tense†, negation†, potential expression, honorific, time
- modification
- SM-subject
- verbal level
- dependency:genitive
- clause head†, clause end†, clause end:adnominal, clause end:complement, clause functional

¹⁶https://github.com/ku-nlp/knp/blob/master/doc/knp_feature.pdf

¹⁷https://github.com/ku-nlp/KWDLc/blob/master/doc/class_feature_manual.pdf

C Training Details

We trained each module with hyper-parameters shown in Table 4. During training, we evaluated a score averaged over tasks on the validation set at the end of each epoch and picked the model with the highest score. When training the word module, the ground-truth word segmentation was used as input. We trained each module three times with different random seeds. Single training runs of the typo, character, and word modules took 38 hours, 2.5 hours, and 5.8 hours on four Tesla V100-SXM2-32GB GPUs, four TITAN X 12GB GPUs, and two Tesla V100-SXM2-32GB GPUs, respectively. The transformers package (Wolf et al., 2020) was used for implementation.

D Single-task Learning Results

Table 5 shows the results of single-task learning. We trained each task in the word module separately in a single-task manner. Note that the training of *POS*, *sub-POS*, *conjugation type*, and *conjugation form* tasks was performed in a multi-task manner as before because these tasks had already achieved enough performance.

Hyper-parameter	Typo Module	Character Module	Word Module
Maximum Sequence Length	256	512	256
Dropout	0.1	0.1	0.1
Batch Size	352	32	16
Maximum Training Epochs	20	20	20
Early Stopping Patience	3	3	3
Warmup Steps	1k	2k	100
Maximum Learning Rate	2e-5	2e-5	1e-4
Learning Rate Decay	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW
AdamW ϵ	1e-6	1e-6	1e-6
AdamW β_1	0.9	0.9	0.9
AdamW β_2	0.99	0.99	0.99
Weight Decay	0.01	0.01	0.01
Gradient Clipping	0.5	0.5	0.5

Table 4: Hyper-parameters for training each module.

Task	Corpus	Metric	KWJA (multi)	KWJA (single)
Morphological Analysis	POS	F1	99.4±0.1	99.4±0.1
	sub-POS	F1	98.7±0.1	98.7±0.0
	conjugation type	F1	99.3±0.3	99.5±0.0
	conjugation form	F1	99.5±0.2	99.6±0.0
	reading	Accuracy	95.8±0.7	96.2±0.0
Named Entity Recognition	all	F1	84.3±4.0	77.9±4.2
Linguistic Feature Tagging	word	F1	98.6±0.1	98.5±0.1
	base phrase	F1	88.3±3.1	92.4±0.1
Dependency Parsing	all	LAS	93.6±0.3	93.5±0.3
PAS Analysis	all	F1	75.9±1.5	79.3±1.0
Bridging Reference Resolution	all	F1	65.8±1.6	65.2±1.6
Coreference Resolution	all	F1	77.7±0.9	77.6±1.2
Discourse Relation Analysis	KWDLC	F1	41.7±0.9	55.3±3.6

Table 5: The performance of KWJA on each task in single-task learning (**single**) compared to that in multi-task learning (**multi**). We fine-tuned KWJA with three different random seeds. We report the mean and standard deviation of the performance. “all” indicates KC, KWDLC, and Fuman corpus, and the metric is the macro-average of them.

Disease Network Constructor: a Pathway Extraction and Visualization

Mohammad Golam Sohrab[†], Khoa N. A. Duong[†], Goran Topic[‡],
Masami Ikeda[†], Nozomi Nagano[†], Yayoi Natsume-Kitatani[‡],
Masakata Kuroda[‡], Mari Nogami Itoh[‡], Hiroya Takamura[†]

[†]Artificial Intelligence Research Center (AIRC),

National Institute of Advanced Industrial Science and Technology (AIST), Japan

[‡]National Institutes of Biomedical Innovation, Health and Nutrition (NIBIOHN), Japan

{sohrab.mohammad, goran.topic, ikeda-masami, takamura.hiroya
n.nagano}@aist.go.jp, {natsume, m-kuroda, mari}@nibiohn.go.jp

Abstract

We present Disease Network Constructor (DNC)¹, a system that extracts and visualizes a disease network, in which nodes are entities such as diseases, proteins, and genes, and edges represent regulation relation. We focused on the disease network derived through regulation events found in scientific articles on idiopathic pulmonary fibrosis (IPF). The front-end web-base user interface of DNC includes two-dimensional (2D) and 3D visualizations of the constructed disease network. The back-end system of DNC includes several natural language processing (NLP) techniques to process biomedical text including BERT-based tokenization on the basis of Bidirectional Encoder Representations from Transformers (BERT), flat and nested named entity recognition (NER), candidate generation and candidate ranking for entity linking (EL) or, relation extraction (RE), and event extraction (EE) tasks. We evaluated the end-to-end EL and end-to-end nested EE systems to determine the DNC's back-end implementation performance. To the best of our knowledge, this is the first attempt that addresses neural NER, EL, RE, and EE tasks in an end-to-end manner that constructs a pathway visualization from events, which we name Disease Network Constructor.

The demonstration video can be accessed from <https://youtu.be/rFhWwAgcXE8>. We release an online system for end users and the source code is available at <https://github.com/aistairc/PRISM-APIs/>.

1 Introduction

In the human body, various substances (entities) such as proteins and compounds interact and regulate each other, forming huge pathway networks.

¹DNC is publicly available at https://biomed-text.airc.aist.go.jp/disease_network/

Such interactions and regulations can be considered as biochemical events. In a disease state, the status of such biochemical events are considered different from those in the healthy state. In order to identify specific substances that can be drug targets in the disease, automatic extraction and visualization of a disease network from scientific articles will be beneficial. The visualization of phenomena and inter-molecular relationships can, for example, make it easier to notice central regulatory molecules, leading to the discovery of drug targets. In this work, we present a system called disease network constructor (DNC) that extracts and visualizes a disease network. We focus on idiopathic pulmonary fibrosis (IPF), which is a severe chronic fibrosis interstitial lung disease, the causes of which remain unclear (Raghu et al., 2011); thus, a deeper understanding of the disease network is urgently needed. DNC is capable of 3D network drawing, and such 3D visualization can help in understanding diseases such as IPF, where complex factors are entangled and multi-level phenomena are involved.

The task formulation of DNC involves several natural language processing (NLP) techniques. DNC is mainly composed of five core models: a Bidirectional Encoder Representation from Transformers (BERT)-based **masked language model** (Devlin et al., 2019), **named entity recognition (NER) model** (Sohrab and Miwa, 2018) that enumerates all possible spans as potential entity mentions and classifies them into entity types, **entity linking (EL) model** (Sohrab et al., 2020a) that executes candidate generation and candidate ranking, **relation extraction (RE) model** (Sohrab et al., 2020b), and **event extraction (EE) model** (Trieu et al., 2020). DNC provides a web-based user interface to facilitate the end-to-end process of neural EL and deep EE on the basis of these five models without any training required by end users. The interface visualizes the 2D and 3D networks on the

basis of output of EL to EE.

2 DNC: Back-end System

The BERT-based back-end system of DNC is built upon four layers:

- NER that uses a contextual neural exhaustive approach to extract mentions, entities, and triggers in text.
- EL that normalizes every detected mention by assigning it an ID in the target knowledge base².
- RE that extracts all possible role pairs (trigger–trigger and trigger–entity pairs) given detected triggers and entities and assigns a role type to each pair.
- EE that enumerates all legal combinations of role pairs to construct event candidates for each trigger.

We employ the modeling of deep EE (Trieu et al., 2020) on the basis of entity, relation, and event over the IPF dataset (Nagano et al., 2023), which is a manually annotated corpus of IPF-related literature. We further extend the end-to-end deep EE model by leveraging the EL (Sohrab et al., 2020a) model to construct a disease network. Figure 1 shows an overview of DNC workflow.

2.1 BERT Layer

To preprocess a given text, we use BERT’s tokenizer to remove special characters and redundant whitespaces, and then split the text into sub-words. A BERT-based pre-trained language model is then used to assign contextual representations to each sub-word.

2.2 Named-entity-recognition Layer

The NER layer assigns entity or trigger types to overlapping text spans by enumerating all possible mention spans on the basis of the same idea as the span-based model (Sohrab and Miwa, 2018; Sohrab et al., 2020b).

2.3 Entity-linking Layer

The EL, or entity normalization, layer, receives the detected mentions $M = \{m_1, m_2, \dots, m_n\}$ from

²<https://www.nlm.nih.gov/research/umls/index.html>

the above NER, where m_i denotes the i -th mention and n denotes the total number of extracted mentions. We address the EL in which detected mentions are mapped to the corresponding concept unique identifiers (CUIs) $C = \{c_1, c_2, \dots, c_n\}$ by leveraging candidate generation and candidate ranking. We use the output of mention extraction as an input to the candidate generation model where we generate a list of k potential CUI candidates for each extracted mention ($k = 50$ in this study). The potential candidates are then fed to the candidate ranking model to select the best candidate for each extracted mention. Our EL layer is based on the EL system of Sohrab et al. (2020a).

2.4 Relation-extraction Layer

The detected mentions and triggers from the NER layer are then fed into the RE layer to assign a role type such as Cause, Cue, Participant, Theme, etc. or no role type to the trigger-argument pairs. The RE layer enumerates all trigger-arguments (trigger-trigger and trigger-entity) to assign a role type.

2.5 Event-extraction Layer

The EE layer receives the detected entities/triggers and the predicted role pairs from the previous layers and enumerates all legal combinations of role pairs to construct event candidates for each trigger. The event candidates include those with and without arguments. Each event candidate is then classified on the basis of whether it is a valid event. Extracting event modifications, such as speculation or negation, is also included in this layer. We describe the event structure to construct event candidates in Section 3.

3 Disease Network Constructor

DNC provides a graph of disease network from the event statistics of IPF. The graph is generated by first applying EL and EE to each input text, then repeatedly collapsing regulation events and their consequents, marking the resultant event with the sign of the regulation event (positive or negative). The resulting graph represents entities as nodes, and regulated events as edges.

We define a “regulation events” as any events with one of the following types: `Positive_regulation`, `Negative_regulation`, or `Regulation`. `Regulation` describes a regulation event for which it is not clear whether its effect is positive

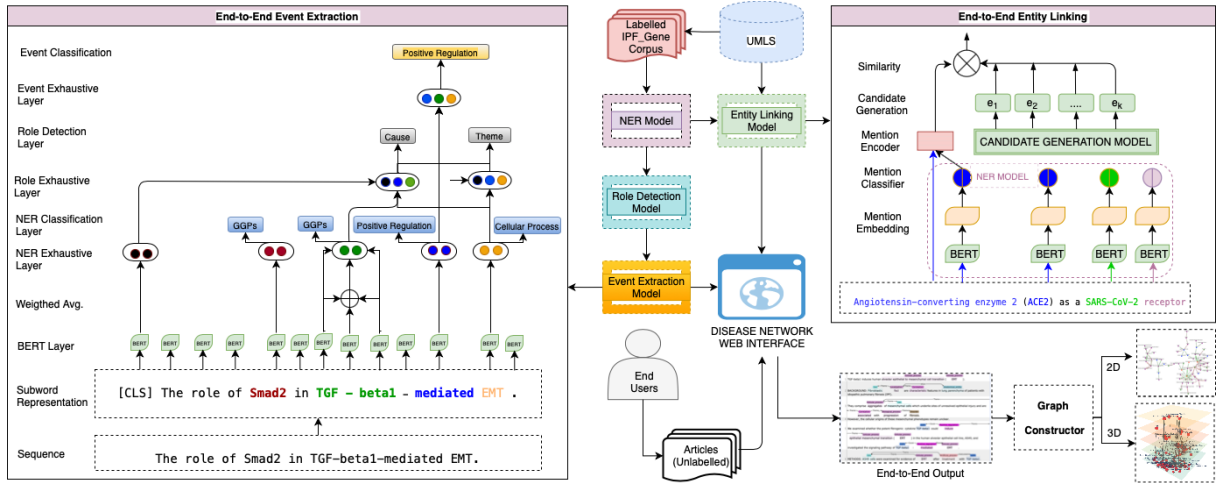


Figure 1: Workflow of DNC.

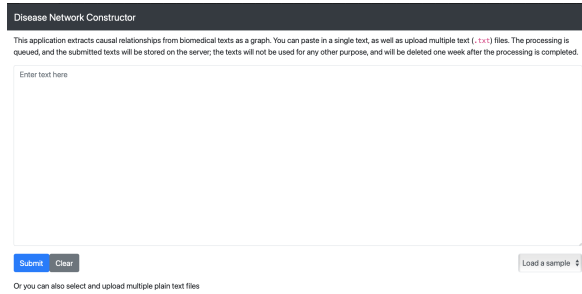


Figure 2: Web-based user interface of DNC.

or negative, and a regulation sign R is defined as $+1$, -1 , and 0 , respectively. A regulation event’s Disorder or Cause roles are considered antecedents, and denoted as A ; Theme roles are consequents, denoted as C . The application collapses regulation events in the following manner.

For each non-negated event E_0 of type T_0 and regulation sign R_0 , for each antecedent-consequent pair (A_0, C_0) where A_0 is an entity (or (null), if E_0 has no antecedents): If C_0 is also an entity, we generate a $\text{Direct_regulation}(A_0, C_0, R_0)$ edge for E_0 . If C_0 is a non-regulation event of type T_1 , for each of its Theme arguments C_1 , we generate a $T_1(A_0, C_1, R_0)$ edge for E_0 . If C_0 is a regulation event, the data (A_0, E_0, C_0) will be remembered as an uncollapsed regulation link.

After each event is processed as above, we iteratively collapse uncollapsed regulation links until it is no longer possible. We look for an uncollapsed regulation link (A_0, E_0, C_0) , such that there exist edges $T_e(A_e, C_e, R_e)$ generated for event C_0 . Those edges are deleted, and new edges $T_e(A_0, C_e, R_0 * R_e)$ is generated for E_0 . Finally,

any edges with a null source is removed.

Intuitively speaking, each regulation event is “folded into” its consequent as its sign. For example, a “negative regulation” of a “gene expression” becomes a “negatively regulated gene expression” edge, connecting the cause of the “negative regulation” with the theme of the “gene expression”. If there is a chain of multiple regulation events, their signs interact: a “negative regulation” of a “negative regulation” of a “gene expression”, for example, becomes a “positively regulated gene expression” edge.

This algorithm results in many events not being included on the graph. Since nodes are entities, any regulation event the antecedents of which are not entities are ignored. Similarly, any regulation events that lacks an antecedent or a consequent, as well as any events that do not have a Theme, will not be included, as there cannot be an edge without both a source and a target node. Any negated events, and any entities that do not participate in at least one edge are also left out.

3.1 DNC: User Interface

Figure 2 shows a user input interface of DNC. For a given text or single or multiple documents from users or a sample text from the provided list, DNC constructs the graph of regulated events on the basis of EL and EE results and visualizes it in 2D and 3D. The user interface also enables visualization of an already pre-computed disease network by uploading an exported .json or .tgz file.

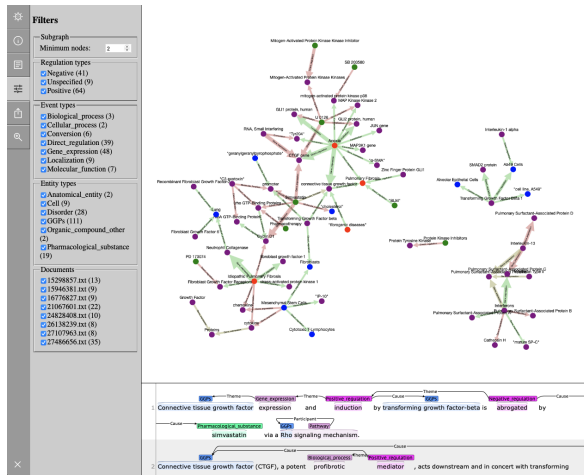


Figure 3: 2D graph produced with DNC.

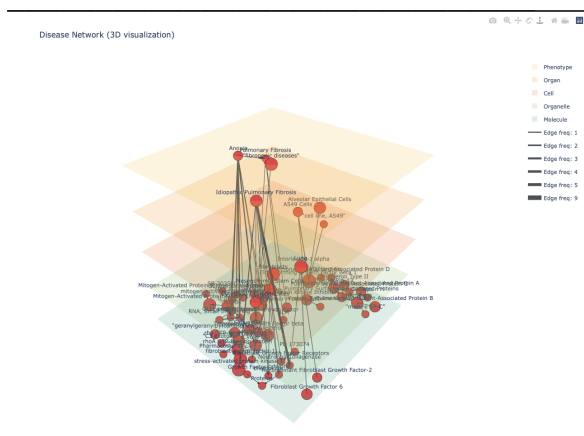


Figure 4: 3D graph produced with DNC.

Information	IPF Dataset
#Documents	150
#Entity types	15
#Event types	13
#Mentions	12868
#Mentions linked to UMLS CUIs	12148
#Mentions linked to NULL	720
#Entities + Triggers	13049
#Relations	6685
#Events	4899
#Modalities	707

Table 1: Statistics of IPF dataset (Nagano et al., 2023).

Data	P	R	F1 (%)
MD	86.12 \pm 1.35	87.13 \pm 1.71	86.61 \pm 1.11
EL	72.54 \pm 2.34	59.44 \pm 3.83	65.30 \pm 2.98

Table 2: EL performance across 10-fold CV on IPF dataset.

3.2 2D Disease Network

The 2D network based on Cytoscape.js³ is used to visualize the 2D network graph. Figure 3 shows a 2D graph of a disease network produced with DNC where ten documents or PubMed abstracts on IPF datae are loaded through the user input interface.

3.2.1 2D Graph Features

In the 2D graph, the edges are colored on the basis of whether they represent a positive or negative regulation, or if the regulation type is unknown. Node colors show their types. Node names are normalized where possible; where EL has not succeeded, mentions are grouped into nodes by literal text of the mention, displayed in quotes. In case the mentions represented by an edge or a node do not all have the same type, the color and/or label reflect that of the majority of mentions.

Besides the standard features of zooming and panning, any edge or node can be selected both for better visibility in the graph and displaying more detailed information in a side panel. Nodes can also be selected from an alphabetical list displayed in the info panel when no selection is made. The info panel also enables the selection of edges and nodes in the neighborhood of the selected element and shows the list of all text mentions that the selected element represents. Clicking on a mention displays the EE result in a brat (Stenetorp et al., 2012) visualization panel, enabling easy checking of the context (see the bottom part of Figure 3).

The graph can be filtered by event and entity types, regulation sign, and by the minimum size of connected subgraphs to be displayed. The zoom button enables quick focus on the selected element as well as an overview of the entire graph. The graph can be exported in several formats: tarball (containing the Cytoscape.js JSON representation of the graph as well as input texts and their EE results, which can be later uploaded to the web app to view without having to re-analyze the documents), JSON only (which can also be uploaded to view, though some features will not be available) of both the full graph and its current filtered state, as well as PNG and SVG images.

3.3 3D Disease Network

The web-base user interface also has a 3D graph function, the main advantage of which is that the nodes are split into layers, representing phenotypes,

³<https://js.cytoscape.org>

Task	10-fold Cross-Validation of End-to-End Entity Linking										
	(%)	Fold ₁	Fold ₂	Fold ₃	Fold ₄	Fold ₅	Fold ₆	Fold ₇	Fold ₈	Fold ₉	Fold ₁₀
MD	P	87.84	87.29	84.19	85.47	87.64	85.54	84.87	85.08	85.57	87.67
	R	88.56	89.17	85.62	86.02	86.57	87.11	87.03	87.94	89.38	83.87
	F	88.21	88.22	84.90	85.75	87.10	86.31	85.93	86.49	87.44	85.73
EL	P	71.96	71.47	75.29	74.31	70.40	68.59	71.52	75.78	74.57	71.48
	R	63.86	61.68	62.84	60.17	53.11	56.66	57.12	59.55	64.29	55.14
	F	67.67	66.22	68.51	66.50	60.54	62.06	63.51	66.69	69.05	62.26

Table 3: EL performances across 10-fold CV. MD indicates mention detection.

Task	P	R	F (%)
NER	84.77 \pm 1.56	82.05 \pm 2.65	83.37 \pm 1.73
RE	58.62 \pm 4.20	59.49 \pm 3.97	58.95 \pm 3.09
EE	51.55 \pm 4.17	40.10 \pm 4.17	45.08 \pm 4.12
ME	51.59 \pm 14.97	26.09 \pm 10.72	34.24 \pm 12.06

Table 4: EE performances across 10-fold of CV on IPF dataset. ME indicates modality extraction.

organs, cells, organelles, and molecules. Figure 4 shows a 3D graph of a disease network produced with DNC.

4 Experimental Settings

In this section, we evaluate the DNC system based on the IPF dataset (Nagano et al., 2023).

4.1 Datasets

We conduct experiments on the IPF dataset which includes 150 abstracts of IPF-related scientific literature where entity, relation, and event information are manually annotated (Nagano et al., 2023). Table 1 shows the statistics of the IPF dataset, which is split into training set and test set for 10-fold cross-validation (CV) in this work. The IPF dataset is randomly divided into 10 folds, each turn, one data fold is used for testing and the remaining folds are used for training.

Moreover, to address IPF-related networks, this dataset includes entity normalization with concept unique identifiers (CUIs) assigned to entities. The UMLS version 2017AA⁴ is used to assign the CUIs of entities. It contains 2.1M unique CUIs which covers 100% of entities in the IPF dataset. As shown in Table 1, the IPF dataset includes 12,319 mentions among which 12,148 and 720

⁴https://www.nlm.nih.gov/pubs/techbull/mj17/mj17_umls_2017aa_release.html

mentions are respectively present and absent in the UMLS. Therefore, the entity coverage ratio of the IPF dataset over the UMLS is around 94.3%.

4.2 Implementation Details

We train the EL and EE models on the pre-trained BERT model and use the pre-trained PubMedBERT (Gu et al., 2020) for end-to-end EL task. We employ the pre-trained SciBERT (Beltagy et al., 2019) model to address the end-to-end event extraction task. We optimize the end-to-end EL and end-to-end EE models using AdamW (Loshchilov and Hutter, 2019) with a learning rate of 3e-5. We train our EL and EE models with 100 epochs and a mini-batch size of 16 on a single graphics processing unit (GPU) with half precision enabled.

5 Results

Table 2 shows the end-to-end EL performances based on the IPF dataset, with the mean scores of precision (P), recall (R), and F-score (F) over the 10-fold CV. The $\pm(\cdot)$ subscript indicates the standard deviation of variation of a set of 10-fold CV scores. Table 3 shows the 10-fold CV end-to-end EL performances over the IPF dataset. The overall performances in Table 2 and the consistent performances over each fold in Table 3, suggest that MD and EL perform well on the IPF dataset.

Table 4 shows the end-to-end EE performances based on the IPF dataset where the end-to-end deep event extraction model is simultaneously trained for entity/trigger, role, and event detection. Since the EE model follows the end-to-end manner, therefore it is noticeable that the model performances are decreased from the NER layer to the modality extraction (ME) layer where each layer error is propagated to the next layer, making the task challenging for the following layers. Table 5 shows the results of the 10-fold CV of the end-to-end deep EE. ME does not perform well compared with the

		10-fold Cross-Validation of End-to-End Deep Event Extraction										
		(%)	Fold ₁	Fold ₂	Fold ₃	Fold ₄	Fold ₅	Fold ₆	Fold ₇	Fold ₈	Fold ₉	Fold ₁₀
NER	P	85.46	84.97	82.74	84.94	88.28	85.82	84.05	84.47	83.49	83.48	
	R	84.01	83.04	81.29	80.71	82.20	82.73	82.55	84.19	84.42	75.35	
	F	84.72	83.99	82.01	82.77	85.13	84.25	83.29	84.33	83.95	79.21	
RE	P	61.13	56.76	60.64	61.10	55.77	62.54	64.53	58.75	54.18	50.83	
	R	62.05	59.95	57.26	57.23	59.61	60.55	60.06	58.83	67.54	51.86	
	F	61.59	58.31	58.91	59.10	57.63	61.53	62.21	58.79	60.12	51.34	
EE	P	52.99	52.19	54.80	50.11	50.93	41.48	52.55	51.42	57.74	51.25	
	R	41.03	39.49	43.51	35.82	39.25	32.56	41.26	43.56	47.01	37.47	
	F	46.24	44.96	48.51	41.78	44.34	36.48	46.22	47.16	51.82	43.29	
ME	P	48.65	64.86	73.08	43.75	58.33	43.75	50.01	19.35	64.10	50.01	
	R	32.14	33.80	32.26	17.28	25.61	18.75	27.85	10.01	46.55	16.67	
	F	38.71	44.44	44.76	24.78	35.59	26.25	35.77	13.19	53.94	25.01	

Table 5: 10-fold cross validation (CV) of end-to-end deep event extraction. ME indicates modality extraction.

other extractions due to insufficient gold data as in Table 1.

6 Related Work

Recent successes in neural networks have shown impressive performance gains on many NLP applications including NER (Lu and Roth, 2015; Ma and Hovy, 2016; Muis and Lu, 2017; Katiyar and Cardie, 2018; Sohrab et al., 2019b; Sohrab and Bhuiyan, 2021), EL (Gupta et al., 2017; Sohrab et al., 2019a), RE (Christopoulou et al., 2019; Jia et al., 2019), and EE (Feng et al., 2016). In contrast, other approaches have emphasized end-to-end EL (Kolitsas et al., 2018), end-to-end RE (Miwa and Bansal, 2016) or even end-to-end EE (Trieu et al., 2020) to facilitate biomedical information extraction tasks. There have been no studies on an all-in-one neural end-to-end approach to facilitate biomedical research, especially to construct disease network pathways that visualize the events along with entity normalization. We addressed this gap by introducing two end-to-end approaches: EL and deep EE to construct a disease network based on IPF, hoping that the DNC can bring insights in making scientific discovery.

Current NLP techniques often use an event representation data format called the “standoff format” to represent their results. Spranger et al. (2015) proposed and discussed a software scheme to convert NLP event representations to standard biomedical pathway data formats (SBML and BioPAX). Apart from neural end-to-end modeling, we integrated brat visualization panels for event representation of the context.

There are several web-based tools exist that support the retrieval of biomedical information using text mining. Sohrab et al. (2020a) introduced BEN-NERD a web-based workflow of NER and EL for NLP research that addresses COVID-19 research. Huang et al. (2021) addressed document-level EE, for extracting entity-centric information such as entity types and entity relations, which is a key to automatic knowledge acquisition from text corpora for various domains. Sohrab et al. (2022) presented an effective web application by addressing entity detection, EL without context using knowledge base application programming interfaces (API), generative RE, and text classification approaches in a pipeline manner for automatic data curation in the biomedical domain. The advantage of this approach is that it can output important fields in a data format that is needed by intended users.

Li et al. (2022) proposed pubmedKB, a web server designed to extract and visualize semantic relationships between four biomedical entity types: variants, genes, diseases, and chemicals. pubmedKB uses state-of-the-art NLP techniques to extract semantic relations from a large number of PubMed abstracts. Deng et al. (2021) addressed an extraction of gene-disease association using a BERT-based language model. Xing et al. (2018) proposed a pipeline based approach to extract the relation between gene-phenotype from biomedical literature.

Many works have shown considerable attention to boost the EE performances. Previous neural models on flat or non-nested EE have been mainly focused on event trigger and argument de-

tection (Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2018; Sha et al., 2018). Besides, deep neural networks including recurrent and convolutional neural networks (CNNs) have boosted EE performance (Björne and Salakoski, 2018; Nguyen and Nguyen, 2019). These models show better performance than traditional hand-crafted feature-based approaches (Björne and Salakoski, 2013; Miwa and Ananiadou, 2013; Yang and Mitchell, 2016). In addition, there are a few end-to-end models (Yang and Mitchell, 2016; Nguyen and Nguyen, 2019) to extract flat events on flat entities; none of these models can treat nested events on nested entities that may further overlap with event triggers. In contrast, Trieu et al. (2020) introduced an end-to-end neural nested EE model which detects nested entities and triggers, roles, nested events; and achieved the new state-of-the-art performance on seven biomedical nested event extraction tasks. In our DNC, we employ the modeling of Deep EE (Trieu et al., 2020) to detect the flat and nested events over the IPF dataset.

7 Conclusion

We present Disease Network Constructor (DNC) to address end-to-end EL and end-to-end deep EE in order to identify and visualize the specific substances (such as proteins) that work differently from those in the healthy state of human bodies. DNC provides an interactive web-based user interface https://biomed-text.airc.aist.go.jp/disease_network/ for enabling real-time visualization and extracting graph information in different formats for end users. We will continue to improve DNC as well as implement new 2D and 3D graph functions to facilitate biomedical research. Moreover, the applicability of this system can be extended to lung diseases such as COVID-19 because some entities and events of the IPF dataset are also related to such diseases.

Acknowledgements

This work is based on results obtained from a project commissioned by the Public/Private R&D Investment Strategic Expansion Program (PRISM). We appreciate insightful feedback from the anonymous reviewers.

References

- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. **SciBERT: A pretrained language model for scientific text**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Jari Björne and Tapio Salakoski. 2013. **TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 shared task**. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25, Sofia, Bulgaria. Association for Computational Linguistics.
- Jari Björne and Tapio Salakoski. 2018. **Biomedical event extraction using convolutional neural networks and dependency parsing**. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.
- Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. **Event extraction via dynamic multi-pooling convolutional neural networks**. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.
- Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. **Connecting the dots: Document-level neural relation extraction with edge-oriented graphs**. 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 4925–4936, Hong Kong, China. Association for Computational Linguistics.
- Chuan Deng, Jiahui Zou, Jingwen Deng, and Mingze Bai. 2021. **Extraction of gene-disease association from literature using biobert**. *The 2nd International Conference on Computing and Data Science*.
- 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.
- Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. **A language-independent neural network for event detection**. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. [Domain-specific language model pretraining for biomedical natural language processing](#).
- Nitish Gupta, Sameer Singh, and Dan Roth. 2017. [Entity linking via joint encoding of types, descriptions, and context](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, Copenhagen, Denmark. Association for Computational Linguistics.
- Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. [Document-level entity-based extraction as template generation](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5257–5269, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Robin Jia, Cliff Wong, and Hoifung Poon. 2019. [Document-level n-ary relation extraction with multi-scale representation learning](#). 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 3693–3704, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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.
- Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. [End-to-end neural entity linking](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.
- Peng-Hsuan Li, Ting-Fu Chen, Jheng-Ying Yu, Shang-Hung Shih, Chan-Hung Su, Yin-Hung Lin, Huai-Kuang Tsai, Hsueh-Fen Juan, Chien-Yu Chen, and Jia-Hsin Huang. 2022. [pubmedKB: an interactive web server for exploring biomedical entity relations in the biomedical literature](#). *Nucleic Acids Research*, 50(W1):W616–W622.
- Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. [Jointly multiple events extraction via attention-based graph information aggregation](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations, ICLR 2019, New Orleans, USA*.
- Wei Lu and Dan Roth. 2015. [Joint mention extraction and classification with mention hypergraphs](#). In *Conference on Empirical Methods in Natural Language Processing*.
- 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.
- Makoto Miwa and Sophia Ananiadou. 2013. [NaCTeM EventMine for BioNLP 2013 CG and PC tasks](#). In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 94–98, Sofia, Bulgaria. Association for Computational Linguistics.
- Makoto Miwa and Mohit Bansal. 2016. [End-to-end relation extraction using LSTMs on sequences and tree structures](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.
- Aldrian Obaja Muis and Wei Lu. 2017. [Labeling gaps between words: Recognizing overlapping mentions with mention separators](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.
- Nozomi Nagano, Narumi Tokunaga, Masami Ikeda, Inoura Hiroko, Duong A. Khoa, Makoto Miwa, Mohammad Golam Sohrab, Goran Topić, Mari Nogami-Itoh, and Hiroya Takamura. 2023. [A novel corpus of molecular to higher-order events that facilitates the understanding of the pathogenic mechanisms of idiopathic pulmonary fibrosis](#). *Scientific Reports*, 13(1, 5986).
- Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. [Joint event extraction via recurrent neural networks](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.
- Trung Minh Nguyen and Thien Huu Nguyen. 2019. [One for all: Neural joint modeling of entities and events](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6851–6858.
- Ganesh Raghunath, Harold R Collard, Jim J Egan, Fernando J Martinez, Juergen Behr, Kevin K Brown, Thomas V Colby, Jean-François Cordier, Kevin R Flaherty, Joseph A Lasky, David A Lynch, Jay H Ryu, Jeffrey J Swigris, Athol U Wells, Julio Ancochea, Demosthenes Bouros, Carlos Carvalho, Ulrich Costabel, Masahito Ebina, David M Hansell, Takeshi Johkoh, Dong Soon Kim, Talmadge E King Jr, Yasuhiro Kondoh, Jeffrey Myers, Nestor L Müller, Andrew G Nicholson, Luca Richeldi, Moisés Selman, Shandra

- L Protzko Holger J Schünemann Rosalind F Dudden, Barbara S Griss, and ATS/ERS/JRS/ALAT Committee on Idiopathic Pulmonary Fibrosis. 2011. [An official ats/ers/jrs/alat statement: idiopathic pulmonary fibrosis: evidence-based guidelines for diagnosis and management](#). *American journal of respiratory and critical care medicine*, 183(6):788–824.
- Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. [Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Mohammad Golam Sohrab and Md. Shoaib Bhuiyan. 2021. [Span-based neural model for multilingual flat and nested named entity recognition](#). In *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, pages 80–84.
- Mohammad Golam Sohrab, Khoa Duong, Makoto Miwa, Goran Topić, Ikeda Masami, and Hiroya Takamura. 2020a. [Bennerd: A neural named entity linking system for covid-19](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 182–188, Online. Association for Computational Linguistics.
- Mohammad Golam Sohrab, Khoa N.A. Duong, Ikeda Masami, Goran Topić, Yayoi Natsume-Kitatani, Masakata Kuroda, Mari Nogami Itoh, and Hiroya Takamura. 2022. [BiomedCurator: Data curation for biomedical literature](#). In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 63–71, Taipei, Taiwan. Association for Computational Linguistics.
- Mohammad Golam Sohrab, Anh-Khoa Duong Nguyen, Makoto Miwa, and Hiroya Takamura. 2020b. [mg-sohrab at WNUT 2020 shared task-1: Neural exhaustive approach for entity and relation recognition over wet lab protocols](#). In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 290–298, Online. Association for Computational Linguistics.
- Mohammad Golam Sohrab and Makoto Miwa. 2018. [Deep exhaustive model for nested named entity recognition](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.
- Mohammad Golam Sohrab, Minh Thang Pham, Makoto Miwa, and Hiroya Takamura. 2019a. [A neural pipeline approach for the PharmaCoNER shared task using contextual exhaustive models](#). In *Proceedings of the 5th Workshop on BioNLP Open Shared Tasks*, pages 47–55, Hong Kong, China. Association for Computational Linguistics.
- Mohammad Golam Sohrab, Pham Minh Thang, and Makoto Miwa. 2019b. [A generic neural exhaustive approach for entity recognition and sensitive span detection](#). In *Proceedings of the Iberian Languages Evaluation Forum co-located with 35th Conference of the Spanish Society for Natural Language Processing, IberLEF@SEPLN 2019, Bilbao, Spain, September 24th, 2019*, volume 2421 of *CEUR Workshop Proceedings*, pages 735–743. CEUR-WS.org.
- Michael Spranger, Sucheendra Palaniappan, and Samik Ghosh. 2015. [Extracting biological pathway models from NLP event representations](#). In *Proceedings of BioNLP 15*, pages 42–51, Beijing, China. Association for Computational Linguistics.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. [brat: a web-based tool for NLP-assisted text annotation](#). In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.
- Hai-Long Trieu, Thy Thy Tran, Khoa N A Duong, Anh Nguyen, Makoto Miwa, and Sophia Ananiadou. 2020. [DeepEventMine: end-to-end neural nested event extraction from biomedical texts](#). *Bioinformatics*, 36(19):4910–4917.
- Wenhui Xing, Junsheng Qi, Xiaohui Yuan, Lin Li, Xiaoyu Zhang, Yuhua Fu, Shengwu Xiong, Lun Hu, and Jing Peng. 2018. [A gene–phenotype relationship extraction pipeline from the biomedical literature using a representation learning approach](#). *Bioinformatics*, 34(13):i386–i394.
- Bishan Yang and Tom M. Mitchell. 2016. [Joint extraction of events and entities within a document context](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

PETALS: Collaborative Inference and Fine-tuning of Large Models

Alexander Borzunov*
HSE University, Yandex

Dmitry Baranchuk*
Yandex

Tim Dettmers*
University of Washington

Max Ryabinin*
HSE University, Yandex

Younes Belkada*
Hugging Face, ENS Paris-Saclay

Artem Chumachenko
Yandex

Pavel Samygin
Yandex School of Data Analysis

Colin Raffel
Hugging Face

Abstract

Many NLP tasks benefit from using large language models (LLMs) that often have more than 100 billion parameters. With the release of BLOOM-176B and OPT-175B, everyone can download pretrained models of this scale. Still, using these models requires high-end hardware unavailable to many researchers. In some cases, LLMs can be used more affordably via RAM offloading or hosted APIs. However, these techniques have innate limitations: offloading is too slow for interactive inference, while APIs are not flexible enough for research that requires access to weights, attention or logits. In this work, we propose PETALS — a system for inference and fine-tuning of large models collaboratively by joining the resources of multiple parties. We demonstrate that this strategy outperforms offloading for very large models, running inference of BLOOM-176B on consumer GPUs with ≈ 1 step per second, which is enough for many interactive LLM applications. Unlike most inference APIs, PETALS also natively exposes hidden states of served models, allowing to train and share custom model extensions based on efficient fine-tuning methods. The system, its source code, and documentation are available at <https://petals.ml>.

1 Introduction

In recent years, the NLP community has found that pretrained language models can solve many practical tasks, through either fine-tuning (Radford et al., 2018) or simple prompting (Brown et al., 2020). Furthermore, performance tends to improve as scale increases (Radford et al., 2019; Kaplan et al., 2020). Following this trend, modern LLMs often have hundreds of billions of parameters (Brown et al., 2020; Rae et al., 2021; Zeng et al., 2021; Kim et al., 2021). Some of these LLMs were released publicly (Zhang et al., 2022; Khrushchev

et al., 2022; Zeng et al., 2022). Most recently, the BigScience project has released BLOOM, a 176 billion parameter model supporting 46 natural and 13 programming languages (Scao et al., 2022).

While the public availability of 100B+ parameter models makes them easier to access, they remain difficult to use for the majority of researchers and practitioners due to memory and computational costs. For instance, OPT-175B and BLOOM-176B need over 350 GB accelerator memory for inference and significantly more for fine-tuning. As a result, these LLMs usually require multiple high-end GPUs or multi-node clusters to be run. Both of these options are extremely expensive, which limits research and potential applications of LLMs.

Several recent works aim to democratize LLMs by “offloading” model parameters to slower but cheaper memory (RAM or SSD), then running them on the accelerator layer by layer (Pudipeddi et al., 2020; Ren et al., 2021). This method allows running LLMs with a single low-end accelerator by loading parameters from RAM just-in-time for each forward pass. Offloading can be efficient for processing many tokens in parallel, but it has inherently high latency: for example, generating one token at a time with BLOOM-176B takes at least 5.5 seconds for the fastest RAM offloading setup and 22 seconds for the fastest SSD offloading. In addition, many computers do not have enough RAM to offload 175B parameters.

Another way to make LLMs more accessible is through public inference APIs, where one party hosts the model and lets others query it over the Internet (OpenAI; AI21; Forefront). Since most of the engineering work is done by the API owner, this is a relatively user-friendly option. However, APIs are often not flexible enough for research use: there is no way to change the model control flow or access internal states. On top of that, current API pricing can make some research projects prohibitively expensive (Liu et al., 2022a).

*Equal contribution. Correspondence to: borzunov.alexander@gmail.com

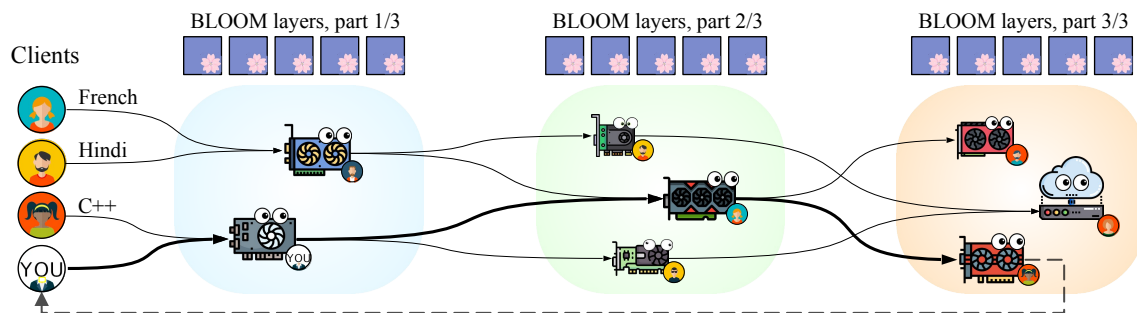


Figure 1: An overview of PETALS. Some participants (*clients*) want to use a pretrained language model to solve various tasks involving processing texts in natural (e.g., French, Hindi) or programming (e.g., C++) languages. They do it with help of other participants (*servers*), who hold various subsets of model layers on their GPUs. Each client chooses a sequence of servers so that it performs an inference or fine-tuning step in the least amount of time.

In this work, we explore an alternative strategy inspired by crowdsourced distributed training of neural networks from scratch (Ryabinin and Gusev, 2020). We introduce PETALS, a platform that allows multiple users to collaborate and perform inference and fine-tuning of large language models over the Internet. Each participant runs a server, a client or both. A *server* hosts a subset of model layers (typically, Transformer blocks) and handles requests from clients. A *client* can form a chain of pipeline-parallel consecutive servers to run the inference of the entire model (Section 2.1). Aside from inference, participants can fine-tune the model through parameter-efficient training methods like adapters (Houlsby et al., 2019) or prompt tuning (Lester et al., 2021) or by training entire layers (Section 2.2). Once trained, submodules can be shared on a model hub (Section 2.3), where others can use them for inference or further training. We demonstrate that existing 100B+ models can run efficiently in this setting with the help of several optimizations: dynamic quantization, prioritizing low-latency connections, and load balancing between servers (Section 3). Finally, we discuss limitations and possible future work (Appendix A).

2 Design and use cases

Practical usage of large language models can be broadly divided into two main scenarios: inference and parameter-efficient adaptation to downstream tasks. In this section, we outline the design of PETALS, showing how it handles both scenarios and also allows easily sharing trained adapters between the users of the system.

2.1 Inference of billion-scale models

When generating tokens, a client stores the model’s token embeddings (which typically comprise a

small fraction of the total parameter count and can fit in RAM in most modern laptops, servers, and workstations) locally and relies on servers to run Transformer blocks. Each server holds several *consecutive* blocks, the number of which depends on the server’s available GPU memory. Before each inference session, the client finds a chain of servers that collectively hold all model layers.

Once the chain is formed, the client uses the local embedding layer to look up embedding vectors for prefix tokens, then sends those vectors to servers and receives new representations. Once the client obtains the outputs of the final block, it computes next token probabilities and repeats this process.

While the session is active, servers store attention keys and values from past client inputs and use them for subsequent inference steps. Clients also store past inputs to each server so that if any server fails or goes offline, another one can quickly take its place. The procedure for finding servers and recovering from failures is detailed in Section 3.2.

Client-side API. To generate tokens with PETALS, one first creates an *inference session*. An inference session iteratively takes inputs as PyTorch tensors, runs them through all Transformer blocks and returns final representations as PyTorch tensors. Under the hood, sessions form server chains, hold cache, and recover from server failures in a way that is transparent to the user. An example of using an inference session is shown in Figure 2.

System requirements. For BLOOM-176B inference, clients need at least 12 GB RAM, most of which is used to store 3.6B embedding parameters. We recommend at least 25 Mbit/s bidirectional bandwidth to avoid bottlenecks in network transfers. Simple greedy inference can use any CPU that runs PyTorch, but more advanced algorithms (e.g., beam search) may require a GPU.

```

# Initialize distributed BLOOM model
model = DistributedBloomForCausalLM \
    .from_pretrained("bigscience/bloom-petals")
input_ids = tokenizer(prefix_text)

with model.inference_session() as session:
    # Session maintains a set of servers that
    # store attention KV from previous steps
    for _ in range(sequence_length):
        # Compute the word embeddings locally
        hid = model.word_embeddings(input_ids)
        # Run distributed Transformer blocks,
        # store attention KV for future steps
        hid = session.step(hid)
        # Sample the next token locally
        probs = model.lm_head(hid)
        input_ids = sample_next_token(probs)

```

Figure 2: A basic PyTorch code snippet for generation with a distributed BLOOM-176B model.

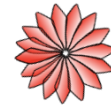
In turn, servers need at least 16 GB of CPU RAM, 100 Mbit/s bandwidth and a GPU with at least 8 GB of memory.

Chat application. We also provide an example application that lets users chat with LLMs in a messenger-like user interface (see Figure 3). The application supports BLOOM-176B and BLOOMZ-176B, a version of BLOOM fine-tuned to better perform in the zero-shot regime (Muenighoff et al., 2022). The application is comprised of the *frontend* and the *backend*. The frontend is a web page that allows users to communicate with the model by prompting it with text and receiving the generated output. The backend is a Flask web server that uses the PETALS client to run inference over the swarm. It accepts requests via HTTP or WebSocket protocols, so anyone can develop their own applications using our backend for inference.

2.2 Training for downstream tasks

While LLMs achieve high quality on many problems with simple prompt engineering (Brown et al., 2020), they often need training to achieve the best results. Traditionally, this is done by fine-tuning all model parameters on the downstream task. However, for very large models, this strategy becomes impractical due to hardware requirements. For example, fine-tuning BLOOM-176B with Adam would require almost 3 TB of GPU memory to store model, gradients, and optimizer states.

To combat this issue, the NLP community has developed *parameter-efficient fine-tuning* methods that keep most of the pretrained model intact. Some of them (Sung et al., 2021; Guo et al., 2021) choose a subset of existing parameters, others (Hu et al., 2021; Houlsby et al., 2019; Liu et al., 2021b; Lester et al., 2021; Liu et al., 2021a, 2022a) augment the



Petals Chat

Human: Hi! I am choosing a name for my new cat, what would you recommend?

AI: Well, in my opinion, a cat's name should be short and easy to pronounce. Why don't you try names between one and three syllables?

Human: |

Figure 3: A chat application that runs BLOOM-176B or BLOOMZ-176B over the PETALS swarm, available at <https://chat.petals.ml>

model with extra trainable weights.

Despite their lower memory requirements, parameter-efficient approaches are often competitive with full model fine-tuning (Hu et al., 2021; Liu et al., 2021a; Yong and Nikoulina, 2022) and even outperform it in low-data regimes (Liu et al., 2022b). Another appealing property of these approaches for our use-case is that they allow rapidly switching a pretrained LLM between different uses.

Distributed fine-tuning. The core principle of fine-tuning in a distributed network is that clients “own” trained parameters while servers host original pretrained layers. Servers can run backpropagation through their layers and return gradients with respect to activations, but they *do not update the server-side parameters*. Thus, clients can simultaneously run different training tasks on the same set of servers without interfering with one another.

To illustrate this principle, we first review an example of soft prompt-tuning for text classification and then generalize it to other methods and tasks. Similarly to Section 2.1, clients store the embedding layers locally and rely on servers to compute the activations of Transformer blocks. In this fine-tuning scenario, a client needs to store trainable soft prompts (task-specific input embeddings) and a linear classification head.

For each training batch, the client routes its data through a chain of remote servers to compute sentence representations, then obtains predictions with the classifier head and computes the cross-entropy


```

# Use distributed BLOOM with soft prompts
model = AutoModelForSequenceClassification \
    .from_pretrained(
        "bigscience/bloom-petals",
        tuning_mode="ptune", pre_seq_len=5)
# Define optimizer for prompts and linear head
opt = torch.optim.AdamW(model.parameters())

for input_ids, labels in data_loader:
    # Forward pass with local & remote layers
    out = model.forward(input_ids)
    loss = cross_entropy(out.logits, labels)

    # Distributed backward w.r.t. local params
    loss.backward() # Compute prompts.grad
    opt.step() # Update local params only
    opt.zero_grad()

```

Figure 4: A basic PyTorch code of soft prompt tuning for sequence classification with PETALS.

loss. During backpropagation, the client runs its data through the same chain of servers in reverse order to compute gradients for the learned prompt vectors. Having obtained those gradients, the client can use a regular PyTorch optimizer to update the parameters of both the head and the prompts, then proceed to the next minibatch.

User interface. To allow users greater flexibility in their training workloads, we made distributed backpropagation module compatible with the PyTorch Autograd engine. Like in the inference stage, this module handles fault tolerance and load balancing transparently to the user while allowing them to access intermediate activations and insert custom PyTorch modules. Figure 4 shows an example training code snippet.

This interface can also support other popular parameter-efficient fine-tuning algorithms, such as LoRA (Hu et al., 2021) or prefix tuning (Li and Liang, 2021). Finally, users can insert custom local modules after some of the existing blocks, which could allow use-cases like retrieval-augmented generation (Borgeaud et al., 2021; Lewis et al., 2020).

2.3 Sharing and reusing trained modules

Although most fine-tuned extensions for pretrained models can be easily shared as-is, simplifying the workflow for sharing these extensions enables users to more easily adapt the model to their target scenario. Indeed, existing model hubs (Wolf et al., 2020; TensorFlow Hub; PyTorch Hub) have gained immense popularity due to many supported models and ease of use, especially when vetting different pretrained models for a given problem. One particularly relevant project is AdapterHub (Pfeiffer et al., 2020), a repository of trained adapters accompanied by a library with implementations of different

adaptation methods. While PETALS does not depend on AdapterHub, it is possible to leverage this library for training adapters in the distributed setting. Instead, we support sharing modules trained by users via the Hugging Face Hub (also used as a backend by AdapterHub). Its infrastructure and the corresponding open source library simplify the learning process for users already familiar with the ecosystem. Because the primary navigation mechanism on the Hugging Face Hub are tags that have been applied to uploaded modules, a user only needs to the task it was trained on and the model upon which the adapter was built. Uploading the weights and the code of the fine-tuned module is done by committing them to a Git repository. When navigating the Hub, users can choose the most suitable adapters by filtering the list of all available modules by the required tags.

3 Internal structure and optimizations

One of the primary considerations for distributed inference is its performance. It can be broken down into three main aspects: computation speed (5-year-old gaming GPU vs. new data center GPU), communication delay due to distance between nodes (intercontinental vs. local), and communication delay due to bandwidth (10 Mbit/s vs. 10 Gbit/s).

In terms of raw FLOPs, even consumer-grade GPUs like GeForce RTX 3070 could run a complete inference step of BLOOM-176B in less than a second (NVIDIA, 2020). However, the GPU memory can only hold a small fraction of model layers: running naively would require 44 RTX 3070 GPUs and 44 communication rounds. To make this more efficient, we use quantization to store more parameters per GPU, reducing the number of consecutive devices and communication rounds (Section 3.1). On top of that, each client prioritizes nearby servers to make communication rounds faster (Section 3.2).

3.1 Large model inference on consumer GPUs

We assume that each server has at least 16 GB of CPU RAM, 8 GB of GPU memory. From this assumption, one of the primary considerations is to reduce the model memory footprint, so that each device can hold more Transformer blocks.

For example, BLOOM has 176B parameters, which takes 352 GB of GPU memory in 16-bit precision. Thus, in the worst case, the model is distributed among 352 GB / 8 GB (per server)

Table 1: Zero-shot accuracy for OPT-175B and BLOOM-176B with 8-bit and 16-bit weights.

Model	Bits	HellaSwag	LAMBADA	WinoGrande	Avg
OPT-175B	16	78.5	74.7	72.6	75.3
	8	78.5	74.6	71.7	74.9
BLOOM	16	73.0	67.2	70.1	70.1
	8	72.8	68.1	70.1	70.3

Table 2: Generation throughput (tokens/s) for BLOOM-176B with 8-bit and 16-bit weights on $8 \times$ A100 GPUs.

Weights	Batch size		
	1	8	32
16-bit	4.18	31.3	100.6
8-bit	3.95	29.4	95.8

= 44 nodes. We can reduce both frequency and amount of data transfer in two ways. First, we can achieve this by compressing the hidden states exchanged between nodes. Second, we can compress the weights to 8-bit precision, reducing the number of nodes required to hold all layers. For BLOOM, this changes the number of required nodes from 44 to 22, which reduces latency in half and decreases the probability of a failure.

Compressing communication buffers. To send less data between subsequent pipeline stages, we use dynamic blockwise quantization (Dettmers et al., 2022b). We apply it to the hidden states before pipeline-parallel communication, as done in Ryabinin et al. (2023). Dynamic blockwise quantization halves the bandwidth requirements without any noticeable effect on generation quality.

Compressing model weights. We use 8-bit mixed matrix decomposition for matrix multiplication to quantize the weights to 8-bit precision and reduce the memory footprint compared to 16-bit weights, as suggested in (Dettmers et al., 2022a). This decomposition separates hidden states and weights into two portions: about 0.1% of 16-bit outlier and 99.9% of 8-bit regular values, which roughly halves the memory footprint.

As shown in Table 1, this method has little effect on LLM quality for major benchmarks. In terms of inference time, Table 2 demonstrates that quantization has about 5% of overhead with batch size 1 (20 tokens), but becomes negligible for larger batches.

3.2 Collaborating over the Internet

Another challenge is to provide *reliable* inference and training despite nodes joining, leaving or failing at any time. To address this, PETALS uses the

hivemind library (Learning@home, 2020) for decentralized training with custom fault-tolerant algorithms for servers and clients detailed below.

Fault-tolerant generation. During inference, clients rely on servers to store attention keys and values for previous tokens. This introduces a potential problem if one or more servers disconnect (or fail) while generating a long sequence. To combat this, PETALS needs a way to recover from server failures transparently to the user.

A naive solution would be to restart the generation procedure, treating previously generated tokens as part of the prompt. This approach has two scaling issues. When generating longer sequences, the inference would have to restart more often, increasing the inference time superlinearly. Also, the more participants take part in the generation procedure, the higher the chance that one of them fails and the entire procedure needs to restart.

To reduce the time spent re-running computations Petals uses a special generation algorithm that supports partial restarts. To enable this, we make both clients and servers store previous activations. While each server stores past keys and values for its local blocks, each client remembers intermediate activations at every “junction” between servers (i.e., the activations it receives from the previous server and sends to the next one).

If one of the servers fail, the client only needs to replace the activations from that server. To do so, the client finds other servers holding the same blocks, then resends the cached activations that were sent to the previous (failed) server. Once this recovery is complete, the replacement server is in the same “inference state” as the rest of the chain, and the client can continue generating tokens.

Communication pattern. The algorithm above implies that clients send requests and receive responses from servers one by one, while servers do not directly pass activations to each other. This is suboptimal for sequential inference, where performance is bounded by the network latency.

To address this, we can make intermediate servers send the output activations both (a) directly to the next server and (b) back to the client. This way, the next server will start computations as soon as possible (after only one network hop instead of two hops), while the client will still be able to reuse the activations in case of server failures. Note that, in this case, sending two times more data does

not worsen performance since, typically, sequential inference is not bounded by network bandwidth.

Server load balancing. First, we ensure that servers are distributed evenly among Transformer blocks. Formally, servers maximize the total model throughput by choosing the blocks with the lowest throughput, thus eliminating potential bottlenecks.

Here, the *block throughput* is the sum of throughputs of all servers hosting this block, while the *server throughput* is the minimum of its network and compute throughputs (in requests/sec), measured empirically before a server joins the system.

Each active server periodically announces its active blocks to a distributed hash table (Maymoukov and Mazieres, 2002). When a new server joins, it uses this information to choose an interval of blocks that contains blocks with the lowest throughput. The server only considers contiguous intervals, since hosting disjointed blocks would harm the inference latency. Once the server selects the best blocks to host, it reports them to the distributed hash table along with its own throughput.

Since peers may leave or fail at any time, all nodes periodically check if launching a rebalancing procedure would significantly improve the overall throughput. If it is the case, they switch layers until the throughput becomes near-optimal. In particular, if all peers serving certain blocks suddenly leave the system, this procedure quickly redistributes the remaining resources to close the emerged gaps.

Client-side routing. Next, we want clients to be able to find a sequence of servers that run the model in the least amount of time. During generation, clients process one or few tokens at a time; in practice, the inference time is mostly sensitive to the network latency. Thus, clients have to ping nearby servers to measure latency and then find the path with minimal time via beam search. Conversely, during fine-tuning one needs to process a batch of examples in parallel. Here, clients can split their batches between multiple servers using the algorithm from Ryabinin et al. (2023). If a server fails, a client removes it from consideration and reruns routing to find a replacement, possibly recovering inference caches as described above.

3.3 Benchmarks

We evaluate the performance of PETALS by running BLOOM-176B in emulated and real-world setups. Our first setup consists of 3 local servers,

Table 3: Performance of sequential inference steps and parallel forward passes. RTT is the round-trip latency.

Network	Single-batch inference (steps/s)		Parallel forward (tokens/s)	
	Sequence length		Batch size	
Bandwidth, RTT	128	2048	1	64
PETALS on 3 physical servers, with one A100 each				
1 Gbit/s, < 5 ms	1.71	1.54	70.0	253.6
100 Mbit/s, < 5 ms	1.66	1.49	56.4	182.0
100 Mbit/s, 100 ms	1.23	1.11	19.7	112.2
PETALS on 12 virtual servers				
1 Gbit/s, < 5 ms	1.24	1.06	37.9	180.0
100 Mbit/s, < 5 ms	1.24	1.05	25.6	66.6
100 Mbit/s, 100 ms	0.57	0.53	5.8	44.3
PETALS on 14 real servers in Europe and North America				
Real world	0.83	0.79	32.6	179.4
Offloading, max. speed on 1x A100				
256 Gbit/s	0.18	0.18	2.7	170.3
128 Gbit/s	0.09	0.09	2.4	152.8
Offloading, max. speed on 3x A100				
256 Gbit/s	0.09	0.09	5.1	325.1
128 Gbit/s	0.05	0.05	3.5	226.3

each running on an A100 80GB GPU. This is an optimistic scenario that requires the least amount of communication. In the second setup, we simulate 12 weaker devices by partitioning each A100-80GB into several virtual servers (3 large and 1 small). We evaluate the above setups with three network configurations: 1 Gbit/s with < 5 ms latency, 100 Mbit/s with < 5 ms latency and 100 Mbit/s with 100 ms latency¹. The client nodes have 8 CPU cores and no GPU.

Next, we benchmark BLOOM in a real-world distributed setting with 14 smaller servers holding 2× RTX 3060, 4× 2080Ti, 2× 3090, 2× A4000, and 4× A5000 GPUs. These are personal servers and servers from university labs, spread across Europe and North America and connected to the Internet at speeds of 100–1000 Mbit/s. Four of the servers operate from under firewalls².

In Table 3, we report the performance of single-batch inference and parallel forward passes for batches of 128-token sequences. For inference, performance does not depend much on bandwidth or sequence length but degrades with higher latency. Parallel forward passes with large batches (used for fine-tuning and parallel inference) are affected by both bandwidth and latency.

¹We simulate network conditions using `tc qdisc`.

²We use the Circuit Relay protocol (libp2p, 2022) to traverse NATs and firewalls.

We also test the effect of having multiple clients. For 12 servers with 100 Mbit/s bandwidth and 100 ms latency, if 8 clients run inference concurrently, each of them gets $\approx 20\%$ slowdown compared to the case when it runs inference alone.

Additionally, we compare PETALS with parameter offloading to run large models with limited resources (Ren et al., 2021; Rajbhandari et al., 2021). For the offloading benchmark we calculate the maximum inference and forward training throughput to receive an upper bound on offloading performance. We base our offloading numbers on the best possible hardware setup for offloading: CPU RAM offloading via PCIe 4.0 with 16 PCIe lanes per GPU and PCIe switches for pairs of GPUs.

We calculate the maximum throughput for offloading as follows. In 8-bit, the model uses 1 GB of memory per billion parameters while PCIe 4.0 with 16 lanes has a throughput of 256 Gbit/s (or 128 Gbit/s if two GPUs are behind a PCIe switch). As such, offloading 176B parameters takes 5.5 seconds for a regular setup and 11 seconds for a multi-GPU setup. We assume an offloading latency of zero for the upper bound estimation.

These results are also shown in Table 3. We can see that offloading is about an order of magnitude slower for single-batch inference compared to PETALS. For the fine-tuning forward pass, offloading is competitive if multiple GPUs are used and the networking for PETALS is limited to 100 Mbit/s or has high latency. In other cases, PETALS offers higher throughput than offloading for training.

4 Conclusion

This paper introduces PETALS, a system for efficient collaborative inference and fine-tuning of large language models. We offer a user-friendly generation interface and a flexible API to access models served over the Internet. We use 8-bit compression that reduces the resource requirements to run very large models. In addition, we develop algorithms for reliable routing and load balancing.

With the release of this system, we hope to broaden access to LLMs and pave the road to applications, studies or research questions that were previously not possible or simply too expensive.

Running LLMs over the Internet raises a broad range of related questions. One of them is privacy: how to avoid revealing private data to outside peers. Another challenge is to ensure that participants can benefit from this system equitably, i.e. in

proportion to their contribution. We discuss future problems such as privacy, security, and incentive structures in Appendix A.

Limitations

An important limitation of our work is *data privacy*: the intermediate activations of the model for given inputs are sent to the servers without any encryption. As such, it might be possible for people hosting the servers to recover the user’s input data. Another limitation is *security*: while there are ways to detect and penalize peers sending faulty outputs, still there is a chance that peers may do that due to faulty hardware or a malicious intent.

Thus, we recommend users working with sensitive data to only use servers hosted by institutions trusted to process this data or set up an isolated PETALS swarm.

We discuss these limitations in more detail in Appendix A and acknowledge that the development of methods for privacy-preserving and secure decentralized inference without performance penalties remains an open research problem.

Ethics Statement

This work introduces a general-purpose algorithm for decentralized inference of large models, aiming to simplify access to the latest research in deep learning. Thus, we do not envision any direct negative impacts from our research aside from granting the broader public an ability to interact with LLMs trained on uncurated web-crawled data. However, all models we serve are already in open access and thus can be exposed via APIs or other means.

Acknowledgements

The authors thank Zheng-Xin Yong, Ilya Dimov, Yozh, Teven Le Scao, Stas Bekman, and Haokun Liu for helpful discussions. We also thank Teven Le Scao for his help in designing Figure 1. A part of the experiments was conducted on a personal server of Elena Voita.

References

- AI21. Jurassic-1 language models. "<https://studio.ai21.com/docs/jurassic1-language-models>". Accessed: 2022-06-22.
- Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. *GPT-NeoX-20B: An open-source autoregressive language model*.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2021. *Improving language models by retrieving from trillions of tokens*.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022a. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *ArXiv*, abs/2208.07339.
- Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022b. 8-bit optimizers via block-wise quantization. *International Conference on Learning Representations (ICLR)*.
- Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathy Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2021. *GLaM: efficient scaling of language models with mixture-of-experts*. *CoRR*, abs/2112.06905.
- David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246.
- Hugging Face and contributors. 2020. Accelerate: Run your raw PyTorch training script on any kind of device. *GitHub*. Note: <https://github.com/huggingface/datasets>, 1.
- William Fedus, Barret Zoph, and Noam Shazeer. 2021. *Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*.
- Forefront. Powerful language models a click away. "<https://www.forefront.ai/>". Accessed: 2022-06-22.
- Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. *A framework for few-shot language model evaluation*.
- Sebastian Gehrmann, Abhik Bhattacharjee, Abinaya Mahendiran, Alex Wang, Alexandros Papangelis, Aman Madaan, Angelina McMillan-Major, Anna Shvets, Ashish Upadhyay, Bingsheng Yao, Bryan Wilie, Chandra Bhagavatula, Chaobin You, Craig Thomson, Cristina Garbacea, Dakuo Wang, Daniel Deutsch, Deyi Xiong, Di Jin, Dimitra Gkatzia, Dragomir Radev, Elizabeth Clark, Esin Durmus, Faisal Ladhak, Filip Ginter, Genta Indra Winata, Hendrik Strobelt, Hiroaki Hayashi, Jekaterina Novikova, Jenna Kanerva, Jenny Chim, Jiawei Zhou, Jordan Clive, Joshua Maynez, João Sedoc, Juraj Juraska, Kaustubh Dhole, Khyathi Raghavi Chandu, Laura Perez-Beltrachini, Leonardo F. R. Ribeiro, Lewis Tunstall, Li Zhang, Mahima Pushkarna, Mathias Creutz, Michael White, Mihir Sanjay Kale, Moussa Kamal Eddine, Nico Daheim, Nishant Subramani, Ondrej Dusek, Paul Pu Liang, Pawan Sasanka Ammanamanchi, Qi Zhu, Ratish Puduppully, Reno Kriz, Rifat Shahriyar, Ronald Cardenas, Saad Mahamood, Salomey Osei, Samuel Cahyawijaya, Sanja Štajner, Sebastien Montella, Shailza, Shailza Jolly, Simon Mille, Tahmid Hasan, Tianhao Shen, Tosin Adewumi, Vikas Raunak, Vipul Raheja, Vitaly Nikolaev, Vivian Tsai, Yacine Jernite, Ying Xu, Yisi Sang, Yixin Liu, and Yufang Hou. 2022. *GEMv2: multilingual NLG benchmarking in a single line of code*.
- Demi Guo, Alexander M Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.
- Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2021. *LoRA: low-rank adaptation of large language models*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. *Scaling laws for neural language models*.
- Michael Khushchhev, Ruslan Vasilev, Nikolay Zinov, Alexey Petrov, and Yandex. 2022. *YaLM 100B*. <https://huggingface.co/yandex/yalm-100b>.

- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. [Dynabench: Rethinking benchmarking in NLP](#).
- Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Dong-Hyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dongpil Seo, Heungsub Lee, Minyoung Jeong, Sungjae Lee, Minsub Kim, SukHyun Ko, Seokhun Kim, Taeyong Park, Jinuk Kim, Soyoung Kang, Na-Hyeon Ryu, Kang Min Yoo, Minsuk Chang, Soobin Suh, Sookyo In, Jinseong Park, Kyungduk Kim, Hiun Kim, Jisu Jeong, Yong Goo Yeo, Donghoon Ham, Dongju Park, Min Young Lee, Jaewook Kang, Inho Kang, Jung-Woo Ha, Woo-Myoung Park, and Nako Sung. 2021. [What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers](#). *CoRR*, abs/2109.04650.
- Team Learning@home. 2020. Hivemind: a Library for Decentralized Deep Learning. <https://github.com/learning-at-home/hivemind>.
- Dmitry Lepikhin, H. Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Y. Huang, M. Krikun, Noam Shazeer, and Z. Chen. 2020. GShard: scaling giant models with conditional computation and automatic sharding. *ArXiv*, abs/2006.16668.
- 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.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- 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.
- libp2p. 2022. libp2p circuit relay. <https://docs.libp2p.io/concepts/nat/circuit-relay/>.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. [Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning](#).
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022b. [Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning](#).
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. [P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks](#). *arXiv preprint arXiv:2110.07602*.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. [GPT understands, too](#). *arXiv:2103.10385*.
- Petar Maymounkov and David Mazières. 2002. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer.
- Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. [Crosslingual generalization through multitask finetuning](#). *arXiv preprint arXiv:2211.01786*.
- Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. [Efficient large-scale language model training on gpu clusters](#). *arXiv preprint arXiv:2104.04473*.
- NVIDIA. 2020. [NVIDIA Ampere GA102 GPU architecture](#).
- NVIDIA. 2022. [NVIDIA confidential computing](#). <https://www.nvidia.com/en-in/data-center/solutions/confidential-computing/>.
- OpenAI. [Build next-gen apps with OpenAI’s powerful models](#). <https://openai.com/api>. Accessed: 2022-06-22.
- Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. [Adapterhub: A framework for adapting transformers](#). *arXiv preprint arXiv:2007.07779*.
- Bharadwaj Pudipeddi, Maral Mesmakhosroshahi, Jinwen Xi, and Sujeeth Bharadwaj. 2020. [Training large neural networks with constant memory using a new execution algorithm](#). *arXiv preprint arXiv:2002.05645*.
- PyTorch Hub. [PyTorch Hub](#). <https://pytorch.org/hub/>. Accessed: 2021-10-04.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. [Improving language understanding by generative pre-training](#).

- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Colin Raffel. 2021. [A call to build models like we build open-source software](#).
- Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14.
- Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. [Zero-offload: Democratizing billion-scale model training](#).
- Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. 2023. SWARM parallelism: Training large models can be surprisingly communication-efficient. *arXiv preprint arXiv:2301.11913*.
- Max Ryabinin and Anton Gusev. 2020. [Towards crowd-sourced training of large neural networks using decentralized mixture-of-experts](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3659–3672. Curran Associates, Inc.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. BLOOM: a 176B-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.
- Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*.
- TensorFlow Hub. TensorFlow Hub. <https://www.tensorflow.org/hub>. Accessed: 2021-10-04.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zheng-Xin Yong and Vassilina Nikoulina. 2022. [Adapting bigscience multilingual model to unseen languages](#).
- Aohan Zeng, Xiao Liu, Zhengxiao Du, Ming Ding, Qinkai Zheng, Hanyu Lai, Zihan Wang, Zhuoyi Yang, Jifan Yu, Xiaohan Zhang, Wendi Zheng, Xiao Xia, Yifan Xu, Weng Lam Tam, Yuxiao Dong, Zixuan Ma, Jiao He, Zhenbo Sun, Jidong Zhai, Wenguang Chen, Guoyang Zeng, Xu Han, Weilin Zhao, Zhiyuan Liu, Yufei Xue, Shan Wang, Jiecai Shan, Haohan Jiang, Zhengang Guo, Peng Zhang, and Jie Tang. 2022. [GLM-130B: An open bilingual pre-trained model](#).
- Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyang Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. [Pangu- \$\alpha\$: Large-scale autoregressive pretrained chinese language models with auto-parallel computation](#). *CoRR*, abs/2104.12369.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. [OPT: open pre-trained transformer language models](#).

Appendix

A Discussion and future work

Incentives for peers to contribute. In PETALS, peers using the client are not required to run a server. This may lead to an imbalance between supply (peers who dedicate GPUs to serve model layers) and demand (peers using the servers to perform inference or fine-tuning for their own needs) in the network. One way to encourage users to serve model layers is to introduce a system of *incentives*: peers running servers would earn special *points*, which can be spent on high-priority inference and fine-tuning or exchanged for other rewards.

Privacy. A key limitation of our approach is that peers serving the first layers of the model can use their inputs to recover input tokens. Thus, clients working with sensitive data should only use the servers hosted by institutions trusted to process this data. This can be achieved with the `allowed_servers` parameter that limits the set of servers a client can use. Alternatively, users can set up their own isolated Petals swarm.

This limitation may be addressed in future work, leveraging the fields of secure multi-party computing (Evans et al., 2018) or privacy-preserving hardware (NVIDIA, 2022).

Security. We assume that servers in our system are run by many independent parties. In practice, some of them may turn out to be faulty and return incorrect outputs instead of the actual results of forward and backward passes. This may happen due to a malicious intent to influence other people’s outputs or, when rewards are introduced (as described above), to earn a reward for serving layers without actually performing the calculations.

A possible way to address these issues would be to use an economically motivated approach. Some servers may vouch for the correctness of their outputs (e.g., in exchange for increased inference price) by depositing a certain number of points as a pledge. Then, for each request, they announce a cryptographic hash of the input and output tensors, so anyone having the inputs can check whether the outputs are correct.

If someone finds a mismatch confirmed by a trusted third party, they can claim the server’s pledge as a reward. In practice, it may be a client who suspects that they received wrong outputs or a “bounty hunter” sending requests to different

servers in the hope of catching errors. While this approach still leaves a chance of receiving wrong outputs, it makes cheating costly and creates an incentive to quickly expose the malicious servers.

Making changes to the main model. As discussed in Section 2.2, distributed parameter-efficient fine-tuning makes it easy for users to apply the base model to new tasks. In Section 2.3, we also described how these updates can be easily shared and reused by others. This capability provides a meaningful step towards *collaborative* improvement of machine learning models (Raffel, 2021): as more and more users train the base model, it will effectively become more capable over time.

Furthermore, we might expect the model parameters that perform best on a specific task to change over time. Similarly to version control systems for code, it would be useful to track versions of fine-tuned model parameters as they change. A system for rapidly testing the performance of a set of parameters on “living benchmarks” (Kiela et al., 2021; Gehrmann et al., 2022; Gao et al., 2021) would be valuable to ensure that subsequent versions improve the desired capabilities.

Apart from adaptation to new tasks, it would also be useful to eventually update the main model. Ideally, such updates could be tracked in a principled way. Users of PETALS could specify the versions of the model they want to use, and servers could indicate which versions they support. Introducing a newer version of the model then reduces to adding a new group of layers, which then naturally supersedes older parameters based on the approach from Section 3.2. Similarly, fine-tuned adapters could be annotated with tags denoting the model version they are applicable for. Such fine-grained model versioning is currently uncommon but would be straightforward to add to PETALS.

UKP-SQuARE v3: A Platform for Multi-Agent QA Research

Haritz Puerto, Tim Baumgärtner, Rachneet Sachdeva, Haishuo Fang, Hao Zhang,
Sewin Tariverdian, Kexin Wang, Iryna Gurevych

Ubiquitous Knowledge Processing Lab (UKP Lab),
Department of Computer Science and Hessian Center for AI (hessian.AI),
Technical University of Darmstadt
www.ukp.tu-darmstadt.de

Abstract

The continuous development of Question Answering (QA) datasets has drawn the research community’s attention toward multi-domain models. A popular approach is to use *multi-dataset models*, which are models trained on multiple datasets to learn their regularities and prevent overfitting to a single dataset. However, with the proliferation of QA models in online repositories such as GitHub or Hugging Face, an alternative is becoming viable. Recent works have demonstrated that combining expert agents can yield large performance gains over multi-dataset models. To ease research in *multi-agent models*, we extend UKP-SQuARE, an online platform for QA research, to support three families of multi-agent systems: i) agent selection, ii) early-fusion of agents, and iii) late-fusion of agents. We conduct experiments to evaluate their inference speed and discuss the performance vs. speed trade-off compared to multi-dataset models. UKP-SQuARE is open-source¹ and publicly available at square.ukp-lab.de.

1 Introduction

The current high-speed development of Artificial Intelligence yields thousands of datasets and trained models in repositories such as GitHub and Hugging Face (Rogers et al., 2023). These models are creating new research and application opportunities, such as high-performing Question Answering (QA) skills in chatbots (Burtsev et al., 2018; Miller et al., 2017). Comparing and analyzing these models usually requires learning libraries, writing code to run the models, and unifying their formats to compare them, which makes this process time-consuming and not scalable.

UKP-SQuARE (Baumgärtner et al., 2022b; Sachdeva et al., 2022) addresses this challenge, providing the first online platform that offers an ecosys-

tem for QA research enabling reproducibility, analysis, and comparison of QA models through a standardized interface and from multiple angles (i.e., general behavior, explainability, adversarial attacks, and behavioral tests).

The large variety of tasks and domains in QA datasets is pushing the research community towards creating models that generalize across domains (Fisch et al., 2019; Talmor and Berant, 2019; Khashabi et al., 2020). Currently, there are two main approaches to achieve this: i) multi-dataset models and ii) multi-agent models. While the former trains a model on multiple datasets (Talmor and Berant, 2019; Khashabi et al., 2020), the latter combines multiple expert agents (Geigle et al., 2021; Friedman et al., 2021; Puerto et al., 2023). Concurrently, large language models (LLM) such as GPT-3 (Brown et al., 2020) are emerging as new powerful systems for multi-task and multi-domain NLP applications. These LLM models are complementary to the focus of our work, multi-agent systems. While LLMs show impressive performance, they are extremely expensive to run and can usually only be accessed through APIs or deployed with great hardware resources. On the other hand, multi-agent systems offer a solution to create multi-domain models reusing available pretrained models that can be run on more modest hardware, which is an important requirement, e.g. where data cannot be sent to third parties.

Multi-agent models are particularly promising due to the thousands of models readily available on online model hubs and their current exponential growth.² This growth in the number of models is increasing the interest of the community in multi-agent model research (Wang et al., 2020; Matena and Raffel, 2021; Geigle et al., 2021; Friedman et al., 2021; Puerto et al., 2023; Wortsman et al., 2022; Jin et al., 2023). However, model hubs such as Hugging Face only allow inference on individ-

¹<https://github.com/UKP-SQuARE/square-core>

²https://www.nazneenrajani.com/emnlp_keynote.pdf

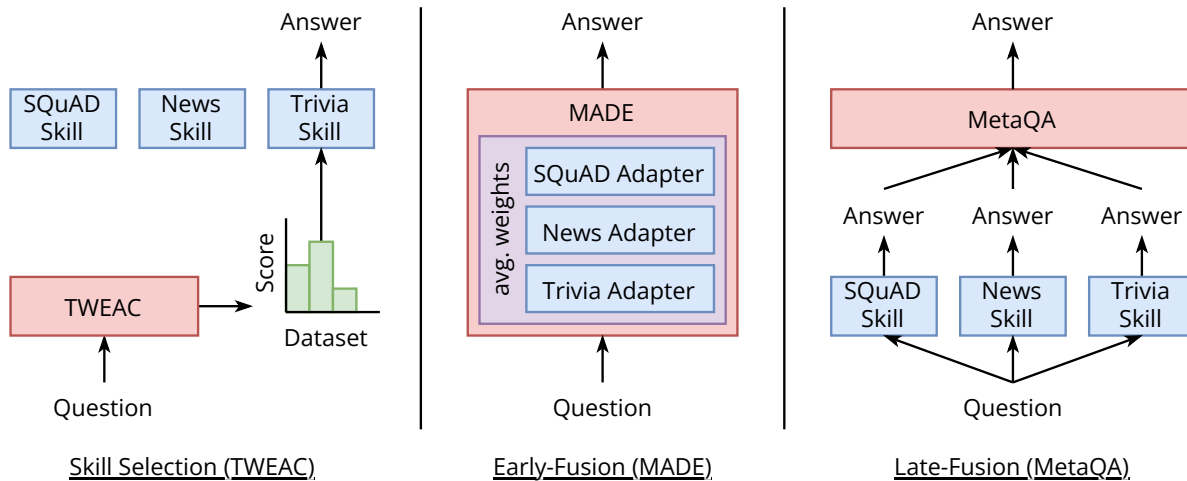


Figure 1: Overview of different multi-agent system architectures deployed in UKP-SQuARE. TWEAC (left) selects an agent (a *Skill* in UKP-SQuARE) based on which dataset it predicts the question is closest to and on which dataset a Skill was trained. MADE (center) fuses the weights of adapters trained on different datasets. MetaQA (right) predicts the final answer from a set of answers and their confidence scores. We illustrate the architectures with three different Skills. However, in practice, more Skills are used.

ual models, disregarding the possibility of combining them to make systems modular and multi-domain. This is a severe limitation as Puerto et al. (2023) showed that combining several QA models can yield performance gains of over 10 percentage points with respect to multi-dataset models (i.e., a single model trained on multiple datasets).

Therefore, we extend UKP-SQuARE to democratize access and research to multi-agent models. In particular, we add support to the three main methods to combine agents³: i) Skill selection, ii) early-fusion of Skills, and iii) late-fusion of Skills. The first consists of identifying the Skill with the highest likelihood of giving the correct answer and then routing the input to that Skill. We deploy TWEAC (Transformer With Extendable QA Agent Classifiers; Geigle et al., 2021) as an example of this method. The second one combines multiple models’ weights to obtain a new model with the distributional knowledge of the source weights. We deploy MADE (Multi-Adapter Dataset Experts; Friedman et al., 2021) as an example of this method. Lastly, the late-fusion of models consists of running multiple models to get their predictions and then combining them. This creates a system that can combine heterogeneous expert agents without reducing their performance in each domain. We provide MetaQA (Puerto et al., 2023) as an example of this method.

UKP-SQuARE facilitates research on multi-

agent QA systems by offering a platform equipped with dozens of agents and three methods to combine them. This upgrade holds paramount significance as the number of QA models created annually is increasing exponentially. UKP-SQuARE enables users to run, compare, and evaluate the strengths and weaknesses of multi-agent models, and compare them with multi-dataset models.

2 Related Work

The most famous types of multi-agent systems are Mixture of Experts (MoE) and ensemble methods. MoE consists of a gating mechanism that routes the input to a set of agents (Jacobs et al., 1991) while ensemble methods aggregate the outputs of multiple experts through a voting mechanism (Breiman, 1996; Freund and Schapire, 1996). Much work has been made to simplify the training of these multi-agent systems (Pedregosa et al., 2011; Chen and Guestrin, 2016; He et al., 2021; Hwang et al., 2022). However, as far as we know, there are no online platforms to run and compare them.

The most similar works to ours are the online model hubs such as Hugging Face’s Model Hub⁴ and AdapterHub (Pfeiffer et al., 2020a). They both offer a large number of models to download. In addition, Hugging Face’s Model Hub also allows running models through Spaces.⁵ However, this re-

³An agent is referred to as *Skill* in UKP-SQuARE.

⁴<https://huggingface.co/models>

⁵<https://huggingface.co/spaces>

quires implementing the Space, which can be non-trivial for complex scenarios such as ours (i.e., deploying and comparing multi-agent systems). UKP-SQuARE removes technical barriers and allows researchers to deploy multi-agent systems with a user-friendly interface.

Transformer (Vaswani et al., 2017) models using adapters (Houlsby et al., 2019) can also be seen as a type of multi-agent system. For this type of architecture, AdapterHub (Pfeiffer et al., 2020a) is a well-established library. In addition to simplifying the training of adapter-based models, it allows composing adapters (i.e., agents) with methods such as AdapterFusion (Pfeiffer et al., 2021) or stacking (Pfeiffer et al., 2020b). However, this library is not an online platform for analyzing models such as UKP-SQuARE. Their focus is to offer tools to create models based on adapters.

3 UKP-SQuARE

UKP-SQuARE (Baumgärtner et al., 2022b; Sachdeva et al., 2022) is the first online platform that offers an ecosystem for QA research. Its goal is to provide a common place to share, run, compare, and analyze QA models from multiple angles, such as explainability, adversarial attacks, behavioral tests, and I/O behaviors. The platform follows a flexible and scalable microservice architecture containing five main services:

- **Datstores:** Provide access to collections of unstructured text such as Wikipedia and Knowledge Graphs such as ConceptNet (Speer and Havasi, 2012).
- **Models:** Enable the dynamic deployment and inference of any Transformer model that implements a Hugging Face pipeline (Wolf et al., 2020) including models that use the adapter-transformers (Pfeiffer et al., 2020a) or sentence-transformers (Reimers and Gurevych, 2019) framework.
- **Skills:** central entity of the UKP-SQuARE. They specify a configurable QA pipeline (e.g., extractive, multiple-choice, and open-domain QA) leveraging Datstores and Models. Users interact with Skills since the platform’s goal is to remove technical barriers and focus on QA research (i.e., the QA pipeline). These Skills are equivalent to agents in the multi-agent system literature.

- **Explainability:** Provides saliency maps, behavioral tests, and graph visualizations⁶ that explains the outputs of a Skill.
- **Adversarial Attacks:** Create modified versions of the input to create adversarial attacks to expose vulnerabilities of the Skills.

All these services allow UKP-SQuARE to offer an ecosystem of tools to analyze Skills through a user-friendly interface without writing any code or complex configurations. UKP-SQuARE helps researchers identify the models’ strengths and weaknesses to push the boundaries of QA research.

3.1 Target Users and Scenarios

This new update of UKP-SQuARE targets researchers working on multi-agent and multi-dataset systems. These users can use the platform as a showcase of their systems. The dozens of Skills already available in UKP-SQuARE simplify the deployment of multi-agent systems since users can employ our user-friendly interface to select the Skills they want to combine using the three families of methods we deploy. Furthermore, researchers can deploy their new multi-skill methods through a pull request in our repository. The platform can also be used to analyze and compare multiple multi-agent systems from efficiency (i.e., inference time) and effectiveness (i.e., performance) points of view. Furthermore, it can also be used to compare multi-agent with multi-dataset systems. Lastly, UKP-SQuARE can also be used for teaching QA. The ecosystem of QA tools can be used to help students understand explainability, adversarial attacks, multi-dataset, and multi-agent models through interactive explanations with examples. Our platform can also be used to design homework where students train QA models and analyze them with the aforementioned QA tools.

4 Multi-Agent Systems

Multi-Agent systems are a type of multi-domain system that aggregate multiple expert agents from different domains to create a unified system. i.e., their focus is on the agents (*Skills* in UKP-SQuARE). On the other hand, multi-dataset systems aim to learn a unified model from multiple data distributions to create a single, general agent. For example, UnifiedQA (Khashabi et al., 2020) is

⁶For graph-based models.

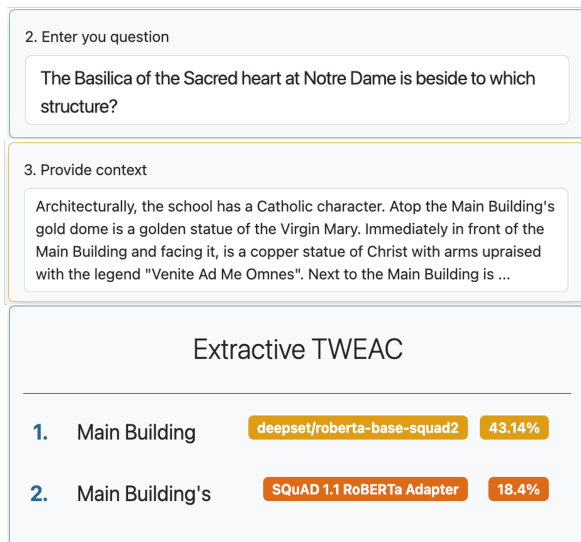


Figure 2: TWEAC predicts that the question is *SQuAD-like* and routes it to Skills trained on this dataset.

a QA model trained on multiple datasets using a generative model to overcome format boundaries.

However, Raffel et al. (2020) show that a model trained on multiple datasets may underperform the same architecture trained on a single dataset, i.e., multi-dataset models may underfit certain distributions. Based on this observation, Puerto et al. (2023) show that multi-agent models can avoid this limitation while being data-efficient to train and even outperform multi-dataset models by large margins in both in-domain and out-of-domain scenarios. This is possible because instead of using a very general architecture to solve multiple tasks, it uses a list of expert agents with specific architectures designed to solve those tasks (i.e., SOTA agents) and establishes a collaboration between these agents. However, this performance comes at a cost. The inference time is higher because it needs to run more than one model (at least one expert agent and one answer aggregator).

Therefore, we extend UKP-SQuARE to add support to the three main approaches for multi-agent systems, which we refer to as Meta-Skills on the platform: i) Skill Selection (§4.1), ii) Early-Fusion of Skills (§4.2), and iii) Late-Fusion of Skills (§4.3). An overview of the different architectures is illustrated in Figure 1.

4.1 Skill Selection

Skill selection is the simplest method of the three. It aims to identify the Skill with the highest likelihood of returning the correct answer to the input question and then route the input to that Skill. More

formally, it defines a function $f : Q \rightarrow S$ that maps any question Q to an available Skill S . Geigle et al. (2021) follow this approach and propose TWEAC (Transformer with Extendable QA Agent Classifiers), a Transformer model with a classification head for each Skill that maps questions to Skills. However, instead of predicting Skills, they predict *datasets*, i.e., they identify the dataset from which the input question comes. Then, they select a Skill trained on that dataset. Using this method, they report a Skill prediction accuracy higher than 90% across ten different QA types.

We train TWEAC on 16 datasets (shown in Appendix 5) with an accuracy of 79% and deploy it in UKP-SQuARE. The cause of the accuracy difference is the selection of the datasets. While the authors experiment on widely different QA tasks such as SQuAD, CommunityQA, and Weather Report, we use the most popular QA datasets, including the 2019 MRQA Shared Task (Fisch et al., 2019), which are more similar and thus, the task becomes more challenging since it is more difficult to distinguish the type of questions. We deploy two TWEAC Skills on UKP-SQuARE: one for extractive QA and another for multiple-choice. Figure 2 shows an extractive QA TWEAC that identifies the question as *SQuAD-like* and routes it to two Skills trained on SQuAD.

4.2 Early-Fusion of Skills

This method combines the weights of multiple models to create a new model that generalizes across all the input models.

Friedman et al. (2021) propose to train adapter weights for individual datasets while sharing the weights of a common Transformer that is also trained with those adapters. Later, in a second training phase, they freeze the Transformer weights and fine-tune each adapter on its corresponding dataset. The intuition behind this is that the shared parameters encode the regularities of the QA task while the adapters model the sub-distributions. This training schema yields a model that performs robustly on new domains by averaging its adapter weights.

Following this work, we extend UKP-SQuARE to allow the creation of Skills that average the weights of a series of adapters. To do this, on the Skill creation page (Figure 3), users are prompted to select whether they wish to combine adapters and, if affirmative, which ones to average.

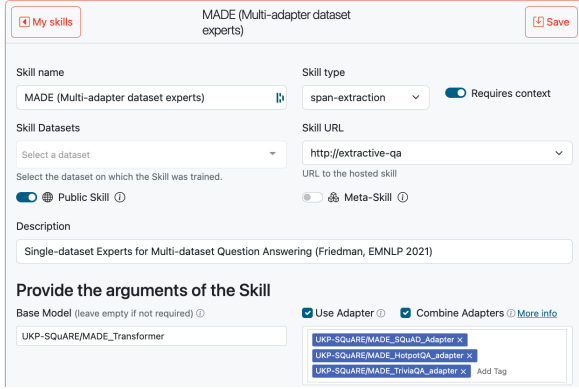


Figure 3: UKP-SQuARE allows combining adapters by simply writing the list of adapters.

4.3 Late-Fusion of Skills

Lastly, Puerto et al. (2023) propose MetaQA, a system that combines 18 heterogeneous expert agents across multiple formats. This system yields significant gains over multi-dataset models because some tasks require particular architectures to solve them, such as DROP (Dua et al., 2019), which requires numerical reasoning. Thus, while a *one-size-fits-all* architecture cannot learn such a wide variety of distributions, a multi-agent system that combines predictions can use expert agents to solve these datasets and yield a higher-performing model in general. Figure 4 shows how MetaQA answers a question from the *DuoRC* dataset but selects an out-of-domain (OOD) agent instead of the in-domain agent to answer, which gives a wrong answer. Thanks to the interface provided by UKP-SQuARE, it is easier to analyze the collaboration between the Skills established by MetaQA.

One limitation of this type of system is its need to run multiple models, which makes it more expensive than the previous two approaches. To alleviate this limitation, we run the expert agents in parallel. In this way, the inference time of MetaQA remains close to the other multi-agent systems, as shown in Table 1.

4.4 Comparison of Multi-Skill Models

In this section, we compare the inference time of the deployed multi-skill systems (i.e., MetaQA, TWEAC, and MADE) and UnifiedQA as a representative of the multi-dataset models. We extract 20 random questions from the six datasets from the MRQA 2019 Shared Task (Fisch et al., 2019) yielding a total of 120 questions and measure the time needed by each Skill to solve them. We repeat this process with five different random seeds and

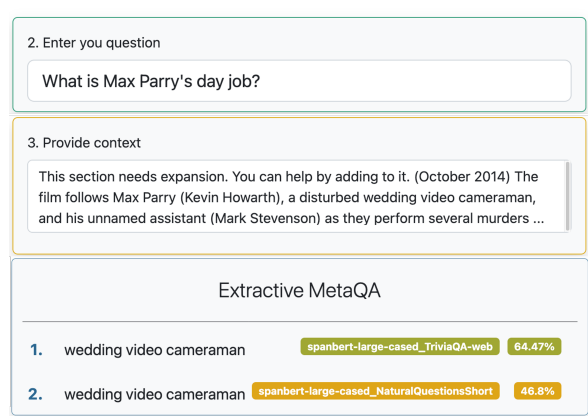


Figure 4: UKP-SQuARE simplifies the analysis of the collaboration between the agents. The question comes from the DuoRC dataset. However, while the in-domain agent gives a wrong answer (not shown), MetaQA selects an out-of-domain agent that gives a correct answer. Only an excerpt of the context is shown.

show the means and standard deviations in Table 1. Each model has 8 CPUs⁷ assigned to it and runs behind an asynchronous API.

As shown in Table 1, MetaQA is the slowest model. This is expected since it needs to run all the expert agents to get the predictions. However, its inference time is remarkably close to both MADE and TWEAC. TWEAC is surprisingly as fast as MADE, considering that TWEAC has to run at least two models (router and expert agent), while MADE only runs one. We conjecture that MADE is not faster because the adapter layers increase the depth of the transformer stack. UnifiedQA is the fastest model, as expected, since it is a multi-dataset model and hence, does not need to combine multiple agents.

Beyond inference, training time and cost are also interesting factors to consider. TWEAC and MetaQA are considered cheap to train assuming the existence of pretrained agents on online model hubs such as the Hugging Face Model Hub.⁸ Hence, the part that they train is a small router or answer aggregator. On the other hand, MADE and UnifiedQA require training a neural network from scratch in the task of question answering, which is much more challenging than simply routing questions or aggregating answers. Therefore, MADE and UnifiedQA need more training data than TWEAC and MetaQA, making them more expensive.

⁷AMD EPYC 7543 with 2.8GHz.

⁸3.6K models on https://huggingface.co/models?pipeline_tag=question-answering&sort=downloads. Accessed on Feb 2023

Model	F ₁	Inference Time [s]	Training
TWEAC	77.65	5.38 ± 0.06	cheap
MADE	82.20	5.45 ± 0.18	expensive
MetaQA	81.13	7.08 ± 0.16	cheap
UnifiedQA	77.30	2.15 ± 0.02	expensive

Table 1: Comparison of inference time on UKP-SQuARE averaged over 600 predictions. Performance from their respective papers.

Table 1 shows the trade-off between performance, training, and inference efficiency. Although MetaQA is the slowest Skill to run, its inference time is very close to the other models’ thanks to the parallel inference of the expert agents offered by UKP-SQuARE (cf. Figure 1). Furthermore, it is cheap to train, has almost the highest performance, and is compatible with any QA format. This makes it interesting for scenarios where model updating, performance, and flexibility are vital. TWEAC is also cheap and as flexible as MetaQA. Although, it is significantly worse than MetaQA on extractive QA datasets. This makes TWEAC ideal in the same scenarios as MetaQA but where running the expert agents in parallel is difficult (i.e., when MetaQA cannot be used). MADE has the highest performance and is as fast as TWEAC. However, it is more expensive to train than MetaQA and TWEAC, and it is not as flexible as MetaQA and TWEAC since it cannot be used for multiple formats simultaneously. Therefore, it should be used when inference, performance, and simple deployment are vital, while the model is not expected to need re-training (i.e., updates) often and is not required to be compatible with multiple QA formats at the same time. Lastly, UnifiedQA is compatible with any text-based QA format but has lower (although competitive) results. Although it is the fastest to run, it is more expensive to train than TWEAC and MetaQA. Thus, its ideal use case is a scenario where a simple deployment is needed while being flexible to process any text-based QA format. Therefore, this small study suggests that in scenarios where new domains are introduced often, router-based systems such as MetaQA might be more suitable, whereas, in scenarios where inference speed or simple deployment are needed, MADE and UnifiedQA might be more appropriate.

5 Conclusions and Discussions

In this work, we have extended UKP-SQuARE to support multi-agent models. In particular, we deployed a routing system, TWEAC (Geigle et al., 2021), a method to combine adapter weights, MADE (Friedman et al., 2021), and a model that combines the prediction of multiple Skills, MetaQA (Puerto et al., 2023). We have conducted experiments on these three models and UnifiedQA (Khashabi et al., 2020), a multi-dataset system, to analyze the trade-off between the performance, efficiency, and flexibility of these systems. We showed that in scenarios where new domains or expertise are often needed, MetaQA provides the best trade-off since its performance is close to the best model, it is compatible with any QA format, cheap to train, and its inference runtime is close to TWEAC and MADE using the parallel engine provided by UKP-SQuARE. However, when simple deployment is needed or the model is not expected to be updated, MADE and UnifiedQA might be more appropriate.

This update of UKP-SQuARE is of utmost importance due to the current speed of development of QA models that creates thousands of models per year. Our platform eases the deployment, running, comparison, and analysis of QA Skills. With this update, we also facilitated the aggregation of these Skills into Multi-Skills simplifying research on multi-agent systems. We leave as future work the comparison of these modular systems with prompting-based QA in large language models (Brown et al., 2020; Zhong et al., 2022).

Limitations

UKP-SQuARE v3 does not aim to provide all existing multi-skill systems off the shelf. Instead, we deploy three different approaches and encourage the community to share, deploy and compare their multi-skill systems. Using the modular Skill system of UKP-SQuARE and the reference implementations, users can reconfigure the existing multi-skill pipelines or implement and deploy their own through a streamlined pull request.⁹

Another limitation is that the multi-skill systems deployed in this paper have been shown to work effectively with no more than 20 Skills. Hence, the effectiveness of multi-skill systems remains unknown for a larger number of Skills. We hope

⁹For details, we refer to the documentation at <https://square.ukp-lab.de/docs/home/components/skills>

that UKP-SQuARE v3 can help shed light on this topic.

Lastly, since multi-skill systems combine several models, it is feasible that the resulting system can inherit biases and unfair behaviors. Although the Skills we used are not intended to exhibit any bias or unfairness, users should use them at their own discretion.

Ethics Statement

Intended Use The intended use of UKP-SQuARE v3 is deploying, running, comparing, analyzing, and combining Skills. Our platform provides dozens of Skills readily available to be combined using the implemented multi-agent systems or new systems to be created by the community. This simplifies the analysis of these systems and thus fosters multi-agent QA research.

Potential Misuse A malicious user could train multiple Skills with biased and unfair behaviors, such as a QA system that responds harmful answers, and combine them with the deployed methods available in UKP-SQuARE. UKP-SQuARE does not provide any Skill with such an intended behavior, but the community is free to upload any model to our platform. Therefore, we encourage the community not to publicly upload such models unless there is a clear research intention with a discussion of the ethics of such research, and in this case, make the Skills private, so that nobody can use them in an unintended way. We are not liable for errors, false, biased, offensive, or any other unintended behavior of the Skills. Users should use them at their own discretion.

Environmental Impact The use of UKP-SQuARE can reduce the computational cost of reproducing prior research since it prevents the community from training models that are already trained.

Acknowledgements

We thank Haris Jabbar, Martin Tutek, and Imbesat Hassan Rizvi for their insightful comments on a previous draft of this paper.

This work has been funded by the German Research Foundation (DFG) as part of the UKP-SQuARE project (grant GU 798/29-1), the QASci-Inf project (GU 798/18-3), and by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research,

Science and the Arts (HMWK) within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

- Tim Baumgärtner, Leonardo F. R. Ribeiro, Nils Reimers, and Iryna Gurevych. 2022a. [Incorporating relevance feedback for information-seeking retrieval using few-shot document re-ranking](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8988–9005, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Tim Baumgärtner, Kexin Wang, Rachneet Sachdeva, Gregor Geigle, Max Eichler, Clifton Poth, Hannah Sterz, Haritz Puerto, Leonardo F. R. Ribeiro, Jonas Pfeiffer, Nils Reimers, Gözde Şahin, and Iryna Gurevych. 2022b. [UKP-SQUARE: An online platform for question answering research](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–22, Dublin, Ireland. Association for Computational Linguistics.
- Leo Breiman. 1996. [Bagging predictors](#). *Mach. Learn.*, 24(2):123–140.
- 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*.
- Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhрева, and Marat Zaynutdinov. 2018. [DeepPavlov: Open-source library for dialogue systems](#). In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.
- Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. [KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base](#). In *Proceedings of the*

- 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.
- Tianqi Chen and Carlos Guestrin. 2016. [Xgboost: A scalable tree boosting system](#). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. [DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs](#). 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.
- Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. [Searchqa: A new q&a dataset augmented with context from a search engine](#). *CoRR*, abs/1704.05179.
- Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. [MRQA 2019 shared task: Evaluating generalization in reading comprehension](#). In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.
- Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 148–156. Morgan Kaufmann.
- Dan Friedman, Ben Dodge, and Danqi Chen. 2021. [Single-dataset experts for multi-dataset question answering](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6128–6137, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Gregor Geigle, Nils Reimers, Andreas Rücklé, and Iryna Gurevych. 2021. [TWEAC: transformer with extendable QA agent classifiers](#). *CoRR*, abs/2104.07081.
- Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. [Fastmoe: A fast mixture-of-expert training system](#). *CoRR*, abs/2103.13262.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for NLP](#). In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.
- Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. 2022. [Tutell: Adaptive mixture-of-experts at scale](#). *CoRR*, abs/2206.03382.
- Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. [Adaptive mixtures of local experts](#). *Neural Comput.*, 3(1):79–87.
- Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. [Dataless knowledge fusion by merging weights of language models](#). In *The Eleventh International Conference on Learning Representations*.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir R. Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2022. [Realtime QA: what’s the answer right now?](#) *CoRR*, abs/2207.13332.
- Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hananeh Hajishirzi. 2020. [UNIFIEDQA: Crossing format boundaries with a single QA system](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- 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.
- Michael Matena and Colin Raffel. 2021. [Merging models with fisher-weighted averaging](#). *CoRR*, abs/2111.09832.

- Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. [ParIAI: A dialog research software platform](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, Copenhagen, Denmark. Association for Computational Linguistics.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. [Scikit-learn: Machine learning in python](#). *J. Mach. Learn. Res.*, 12:2825–2830.
- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. [AdapterFusion: Non-destructive task composition for transfer learning](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.
- Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. [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.
- Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. [MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.
- Haritz Puerto, Gözde Şahin, and Iryna Gurevych. 2023. [MetaQA: Combining expert agents for multi-skill question answering](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3566–3580, Dubrovnik, Croatia. 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](#). *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- 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.
- Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. [QA dataset explosion: A taxonomy of NLP resources for question answering and reading comprehension](#). *ACM Comput. Surv.*, 55(10):197:1–197:45.
- Rachneet Sachdeva, Haritz Puerto, Tim Baumgärtner, Sewin Tariverdian, Hao Zhang, Kexin Wang, Hosain Shaikh Saadi, Leonardo F. R. Ribeiro, and Iryna Gurevych. 2022. [UKP-SQuARE v2: Explainability and adversarial attacks for trustworthy QA](#). In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 28–38, Taipei, Taiwan. Association for Computational Linguistics.
- Robyn Speer and Catherine Havasi. 2012. [Representing general relational knowledge in ConceptNet 5](#). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).
- Alon Talmor and Jonathan Berant. 2019. [MultiQA: An empirical investigation of generalization and transfer in reading comprehension](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.
- Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordani, Philip Bachman, and Kaheer Suleman. 2017. [NewsQA: A machine comprehension dataset](#). In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. 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.
- Jesse Vig. 2019. [A multiscale visualization of attention in the transformer model](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.
- Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: a free collaborative knowledgebase](#). *Commun. ACM*, 57(10):78–85.

- Jing Wang, Mayank Kulkarni, and Daniel Preotiuc-Pietro. 2020. [Multi-domain named entity recognition with genre-aware and agnostic inference](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8476–8488, Online. 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.
- Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. [Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time](#). In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Wanjuan Zhong, Yifan Gao, Ning Ding, Yujia Qin, Zhiyuan Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022. [ProQA: Structural prompt-based pre-training for unified question answering](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4230–4243, Seattle, United States. Association for Computational Linguistics.

A Further Updates

Although our focus is the development of multi-domain systems, we further extended UKP-SQuARE with other minor but important features.

A.1 Knowledge Graph Question Answering

To maximize the advantage of multi-agent systems, UKP-SQuARE needs to be compatible with most QA systems available. UKP-SQuARE is already compatible with most QA formats (i.e., extractive, multiple-choice, boolean, and abstractive), and in this version, we add support for Knowledge Graph Question Answering (KGQA) systems. Sachdeva et al. (2022) include support for neuro-symbolic systems that combine language models with ConceptNet but lack support for KGQA models. Thus, this paper implements a generic KGQA Skill compatible with any knowledge graph and any generation model that generates SPARQL queries. As a demo, we deploy a BART-based (Lewis et al., 2020) semantic parser on KQA Pro (Cao et al., 2022), which is a complex KGQA dataset based on Wikidata (Vrandečić and Krötzsch, 2014) with nine different question types. The KGQA Skill parses a question into an executable SPARQL query, which is then executed against a KG to get the final answer. For this purpose, a dataset-centric subgraph is deployed using virtuoso.¹⁰ Thanks to the modularity of UKP-SQuARE, we can flexibly combine different semantic parsers with different KGs to get the final answer. Therefore, all aforementioned multi-Skill methods can be easily adapted for multi-Skill KGQA, which we leave as future work.

A.2 BERTViz

We also extended our explainability ecosystem by adding BERTViz (Vig, 2019), a method that allows the exploration of the attention weights as shown in Figure 5. While UKP-SQuARE v2 focuses on high-level explanations through saliency maps, BERTViz offers a low-level explanation utilizing the attention weights across all layers of the transformer models.

A.3 Datastores

Lastly, regarding Datastores, while UKP-SQuARE v1 focuses on document collections such as Wikipedia or PubMed, and UKP-SQuARE v2 focuses on Knowledge Graphs, UKP-SQuARE v3 offers a datastore that is updated in real-time. This

type of information is vital for real-time questions (i.e., questions whose answers may change over time; Kasai et al. (2022)). For instance, some facts, such as the president of a country, can change quickly. Therefore, we deploy a real-time datastore by using the Bing Search API. This datastore does not store documents and, instead, relies on the Bing Search engine to retrieve online documents (i.e., websites) that are more likely to be updated.

Furthermore, we create an information-retrieval Skill that allows the inference of only an IR model (instead of combining them with a reader model). We allow providing relevance-feedback for sparse retrieval, as it has shown to perform well in information-seeking and interactive scenarios (Baumgärtner et al., 2022a).

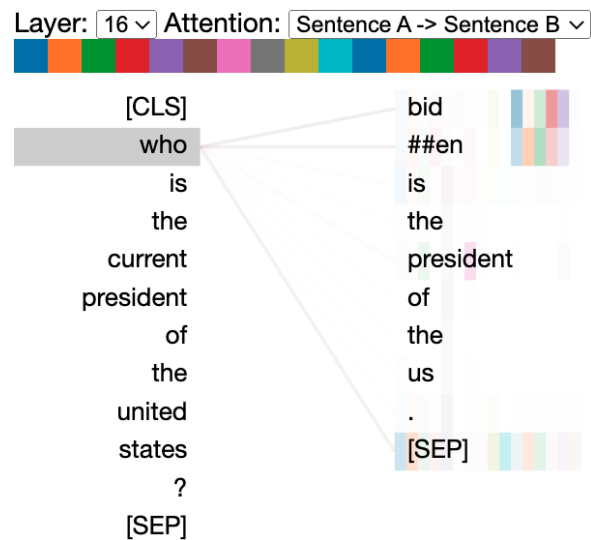


Figure 5: Screenshot of the BERTViz attention visualization in UKP-SQuARE. The token *who* attends to *Biden* (the answer) more than to other tokens.

¹⁰<https://virtuoso.openlinksw.com/>

B Datasets

Dataset	Characteristics	Train	Dev	Test	License
SQuAD (Rajpurkar et al., 2016)	Crowdsourced questions on Wikipedia	6573	5253	5254	MIT
NewsQA (Trischler et al., 2017)	Crowdsourced questions about News	74160	2106	2106	MIT
HotpotQA (Yang et al., 2018)	Crowdsourced multi-hop questions on Wikipedia	72928	2950	2951	MIT
SearchQA (Dunn et al., 2017)	Web Snippets, Trivia questions from J! Archive	117384	8490	8490	MIT
NQ (Kwiatkowski et al., 2019)	Wikipedia, real user queries on Google Search	104071	6418	6418	MIT
TriviaQA-web (Joshi et al., 2017)	Web Snippets, crowdsourced trivia questions	61688	3892	3893	MIT

Table 2: Summary of the datasets used in §4.4.

C Expert Agents

Expert Agents	Link
Span-BERT Large for SQuAD	https://huggingface.co/haritzpuerto/spanbert-large-cased_SQuAD
Span-BERT Large for NewsQA	https://huggingface.co/haritzpuerto/spanbert-large-cased_NewsQA
Span-BERT Large for HotpotQA	https://huggingface.co/haritzpuerto/spanbert-large-cased_HotpotQA
Span-BERT Large for SearchQA	https://huggingface.co/haritzpuerto/spanbert-large-cased_SearchQA
Span-BERT Large for NQ	https://huggingface.co/haritzpuerto/spanbert-large-cased_NaturalQuestionsShort
Span-BERT Large for TriviaQA-web	https://huggingface.co/haritzpuerto/spanbert-large-cased_TriviaQA-web
Span-BERT Large for QAMR	https://huggingface.co/haritzpuerto/spanbert-large-cased_QAMR
Span-BERT Large for DuoRC	https://huggingface.co/haritzpuerto/spanbert-large-cased_DuoRC

Table 3: List of the expert agents used for TWEAC and MetaQA.

Ranger: A Toolkit for Effect-Size Based Multi-Task Evaluation

Mete Sertkan*
CD-Lab RecSys @ TU Wien
mete.sertkan@tuwien.ac.at

Sophia Althammer*
TU Wien
sophia.althammer@tuwien.ac.at

Sebastian Hofstätter*
Cohere
s.hofstaetter@tuwien.ac.at

Abstract

In this paper, we introduce *Ranger* - a toolkit to simplify the utilization of effect-size-based meta-analysis for multi-task evaluation in NLP and IR. We observed that our communities often face the challenge of aggregating results over incomparable metrics and scenarios, which makes conclusions and take-away messages less reliable. With *Ranger*, we aim to address this issue by providing a task-agnostic toolkit that combines the effect of a treatment on multiple tasks into one statistical evaluation, allowing for comparison of metrics and computation of an overall summary effect. Our toolkit produces publication-ready forest plots that enable clear communication of evaluation results over multiple tasks. Our goal with the ready-to-use *Ranger* toolkit is to promote robust, effect-size based evaluation and improve evaluation standards in the community. We provide two case studies for common IR and NLP settings to highlight *Ranger*'s benefits.

1 Introduction

We in the NLP (natural language processing) and IR (information retrieval) communities maneuvered ourselves into somewhat of a predicament: We want to evaluate our models on a range of different tasks to make sure they are robust and generalize well. However, this goal is often reached by aggregating results over incomparable metrics and scenarios (Thakur et al., 2021; Bowman and Dahl, 2021). This in turn makes conclusions and take away messages much less reliable than we would like. Other disciplines, such as social and medical sciences have much more robust tools and norms in place to address the challenge of meta-analysis.

In this paper we present *Ranger* – a toolkit to facilitate an easy use of effect-size based meta-analysis for multi-task evaluation. *Ranger* produces beautiful, publication-ready forest plots to

help everyone in the community to clearly communicate evaluation results over multiple tasks. *Ranger* is written in python and makes use of matplotlib. Thus it will be easy and time-efficient to customize if needed.

With the effect-size based meta-analysis (Borenstein et al., 2009) *Ranger* lets you synthesize the effect of a treatment on multiple tasks into one statistical evaluation. Since in meta-analysis the influence of each task on the overall effect is measured with the tasks' effect size, meta-analysis provides a robust evaluation for a suite of tasks with more insights about the influence of one task for the overall benchmark. With the effect-size based meta-analysis in *Ranger* one can compare metrics across different tasks which are not comparable over different test sets, like nDCG, where the mean over different test sets holds no meaning. *Ranger* is not limited to one metric and can be used for all evaluation tasks with metrics, which provide a sample-wise metric for each sample in the test set. *Ranger* can compare effects of treatments across different metrics. How the effect size is measured, depends on experiment characteristics like the computation of the metrics or the homogeneity of the metrics between the multiple tasks. In order to make *Ranger* applicable to a wide range of multi-task evaluation, *Ranger* offers effect size measurement using the mean differences, the standardized mean difference or the correlation of the metrics. In order to have an aggregated, robust comparison over the whole benchmark, *Ranger* computes an overall combined summary effect for the multi-task evaluation. Since these statistical analysis are rather hard to interpret by only looking at the numbers, *Ranger* includes clear visualization of the meta-analysis comprised in a forest plot as in Figure 1.

In order to promote robust, effect-size based evaluation of multi-task benchmarks we open source the ready-to-use toolkit at:
<https://github.com/MeteSertkan/ranger>

*All authors contributed equally

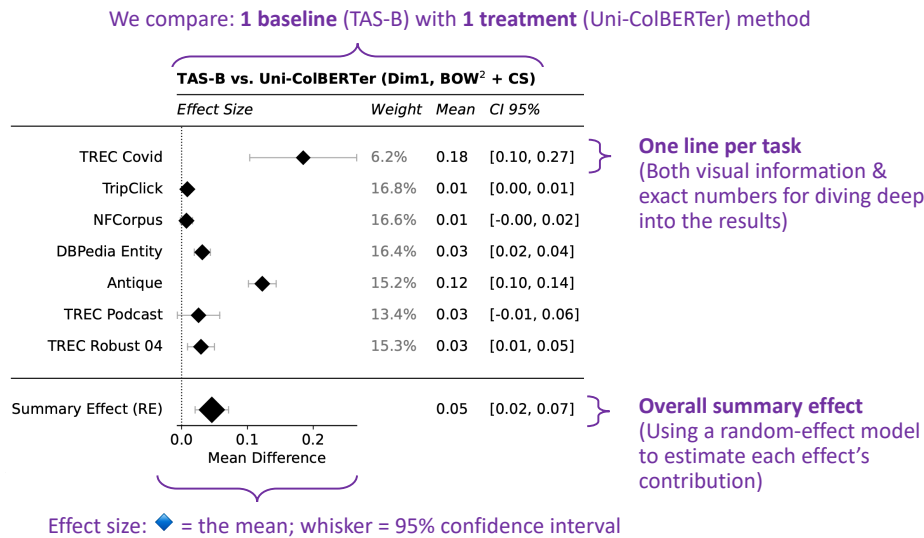


Figure 1: Example forest plot, with explanations highlighting the output of our *Ranger* toolkit for a multi-task meta-analysis using effect sizes between a control baseline and treatment methods (In this case we use experiments from ColBERTer (Hofstätter et al., 2022)).

2 Related Work

In the last years, increasingly more issues of benchmarking have been discussed in NLP (Church et al., 2021; Colombo et al., 2022) and IR (Craswell et al., 2022; Voorhees and Roberts, 2021). Bowman and Dahl (2021) raise the issue that unreliable and biased systems score disproportionately high on benchmarks in Natural Language Understanding and that constructing adversarial, out-of-distribution test sets also only hides the abilities that the benchmarks should measure. Bowman (2022) notice that the now common practices for evaluation lead to unreliable, unrealistically positive claims about the systems. In IR one common multi-task benchmark is BEIR Thakur et al. (2021), which evaluates retrieval models on multiple tasks and domains, however in the evaluation the overall effect of a model is measured by averaging the nDCG scores over each task. As the nDCG score is task dependent and can only be compared within one task, it can not be averaged over different tasks and the mean of the nDCG scores does not hold any meaning. Thus there is the urgent need in the NLP and IR community for a robust, synthesized statistical evaluation over multiple tasks, which is able to aggregate scores, which are not comparable, to an overall effect.

To address these needs Soboroff (2018) propose

effect-size based meta-analysis with the use case of evaluation in IR for a robust, aggregated evaluation over multiple tasks. Similarly, Colombo et al. (2022) propose a new method for ranking systems based on their performance across different tasks based on theories of social choice.

In NLP and IR research exist numerous single evaluation tools (Azzopardi et al., 2019; MacAvaney et al., 2022), however to the best of our knowledge there exists no evaluation tool addressing the needs for robust, synthesized multi-task evaluation based on meta-analysis.

In order to make this robust, effect-size based evaluation easily accessible for a broad range of tasks, we present our *Ranger* toolkit and demonstrate use cases in NLP and IR.

3 Ranger

3.1 Methodology

Besides analyzing the effects in individual studies, meta-analysis aims to summarize those effects in one statistical synthesis (Borenstein et al., 2009). Translated to the use case of NLP and IR, meta-analysis is a tool to compare whether a treatment model yields gains over a control model within different data collections and overall (Soboroff, 2018). A treatment, for example, could be an incremental update to the control model, a new model, or a

model trained with additional data; a control model can be considered as the baseline (e.g., current SOTA, etc.) to compare the treatment with. To conduct a meta-analysis, defining an effect size is necessary. In this work, we quantify the effect size utilizing the raw mean difference, the standardized mean difference, and the correlation. In particular, we implement the definitions of those effect sizes as defined by [Borenstein et al. \(2009\)](#) for paired study designs since, typically, the compared metrics in IR and NLP experiments are obtained by employing treatment and control models on the same collections.

Raw Mean Difference D . In IR and NLP experiments, researchers usually obtain performance metrics for every item in a collection. By comparing the average of these metrics, they can make statements about the relative performance of different models. Thus, the difference in means is a simple and easy-to-interpret measure of the effect size, as it is on the same scale as the underlying metric. We compute the raw mean difference D by averaging the pairwise differences between treatment X_T and control metric X_C and use the standard deviation (S_{diff}) of the pairwise differences to compute its corresponding variance V_D as follows:

$$D = \frac{X_T - X_C}{n}, \quad (1)$$

$$V_D = \frac{S_{diff}^2}{n},$$

where n is the number of compared pairs.

Standardized Mean Difference d . Sometimes, we might consider standardizing the mean difference (i.e., transforming it into a “unitless” form) to make the effect size comparable and combinable across studies. For example, if a benchmark computes accuracy differently in its individual collections or employs different ranking metrics. The standardized mean difference is computed by dividing the raw mean difference D by the within-group standard deviation S_{within} calculated across the treatment and control metrics.

$$d = \frac{D}{S_{within}} \quad (2)$$

Having the standard deviation of the pairwise differences S_{diff} and the correlation of the corresponding pairs r , we compute S_{within} as follows:

$$S_{within} = \frac{S_{diff}}{\sqrt{2(1-r)}} \quad (3)$$

The variance of standardized mean difference d is

$$V_d = \left(\frac{1}{n} + \frac{d^2}{2n}\right)2(1-r), \quad (4)$$

where n is the number of compared pairs. In small samples, d tends to overestimate the absolute value of the true standardized mean difference δ , which can be corrected by factor J to obtain an unbiased estimate called Hedges’ g ([Hedges, 1981](#); [Borenstein et al., 2009](#)) and its corresponding variance V_g :

$$J = 1 - \frac{3}{4df - 1},$$

$$g = J \times d, \quad (5)$$

$$V_g = J^2 \times V_d,$$

where df is degrees of freedom which is $n - 1$ in the paired study setting with n number of pairs.

Correlation r . Some studies might utilize the correlation coefficient as an evaluation metric, for example, how the output of an introduced model (treatment) correlates with a certain gold standard (control). In such cases, the correlation coefficient itself can serve as the effect size, and its variance is approximated as follows:

$$V_r = \frac{(1-r^2)^2}{n-1}, \quad (6)$$

where n is the sample size. Since the variance strongly depends on the correlation, the correlation coefficient is typically converted to Fisher’s z scale to conduct a meta-analysis ([Borenstein et al., 2009](#)). The transformation and corresponding variance is:

$$z = 0.5 \times \ln\left(\frac{1+r}{1-r}\right), \quad (7)$$

$$V_z = \frac{1}{n-3}$$

As already mentioned, z and V_z are used throughout the meta-analysis; however, for reporting/communication, z metrics are transformed back into the correlation scale using:

$$r = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (8)$$

Combined Effect M^* . After calculating the individual effect sizes (Y_i) and corresponding variances (V_{Y_i}) for a group of k experiments, the final step in meta-analysis is to merge them into a single summary effect. As [Soboroff \(2018\)](#), we assume heterogeneity, i.e., that the effect size variance

varies across the experiments. Following (Sobroff, 2018), we employ the random-effects model as defined in (Borenstein et al., 2009) to consider the between-study variance T^2 for the summary effect computation. We use the DerSimonian and Laird method (DerSimonian and Laird, 2015) to estimate T^2 :

$$\begin{aligned}
 T^2 &= \frac{Q - df}{C}, \\
 Q &= \sum_{i=1}^k W_i Y_i^2 - \frac{(\sum_{i=1}^k W_i Y_i)^2}{\sum_{i=1}^k W_i}, \\
 df &= k - 1, \\
 C &= \sum W_i - \frac{\sum W_i^2}{\sum W_i}.
 \end{aligned} \tag{9}$$

where the weight of the individual experiments $W_i = 1/V_{Y_i}$. We adjust the weights by T^2 and compute the weighted average of the individual effect sizes, i.e., the summary effect M^* , and its corresponding variance V_{M^*} as follows:

$$\begin{aligned}
 W_i^* &= \frac{1}{V_{Y_i} + T^2}, \\
 M^* &= \frac{\sum_{i=1}^k W_i^* Y_i}{\sum_{i=1}^k W_i^*}, \\
 V_{M^*} &= \frac{1}{\sum_{i=1}^k W_i^*}.
 \end{aligned} \tag{10}$$

Confidence Interval (CI). We determine the corresponding confidence interval (represented by the lower limit, LL_Y , and the upper limit, UL_Y) for a given effect size Y , which can be the result of an individual experiment (Y_i) or the summary effect (M^*), as follows:

$$\begin{aligned}
 SE_Y &= \sqrt{V_Y}, \\
 LL_Y &= Y - Z^\alpha \times SE_Y, \\
 UL_Y &= Y + Z^\alpha \times SE_Y,
 \end{aligned} \tag{11}$$

where SE_Y is the standard error, V_Y the variance of the effect size, and Z^α the Z-value corresponding to the desired significance level α . Given α we compute Z^α :

$$Z^\alpha = ppf(1 - \frac{\alpha}{2}), \tag{12}$$

where $ppf()$ is the percent point function (we use `scipy.stats.norm.ppf1`). For example, $\alpha = 0.05$ yields the 95% CI of $Y \pm 1.96 \times SE_Y$.

¹<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html>

Forest Plots. The meta-analysis results in the individual experiments' effect sizes, a statistical synthesis of them, and their corresponding confidence intervals. Forest plots are a convenient way of reporting those results, which enables a very intuitive interpretation at one glance. *Ranger* supports forest plots out of the box, which can easily be customized to one's needs since it is based on python and matplotlib. We provide an example with explanations in Figure 1. Effect sizes and corresponding confidence intervals are depicted as diamonds with whiskers $\vdash \blacklozenge \dashv$. The size of the diamonds is scaled by the experiments' weights (W_i^*).

The dotted vertical line $\dot{}$ at zero represents the zero effect. The observed effect size is not significant when its confidence interval crosses the zero effect line; in other words, we cannot detect the effect size at the given confidence level.

3.2 Usage

We explain the easy usage of *Ranger* along with two examples on classification evaluation of GLUE in NLP and retrieval evaluation of BEIR in IR.

The meta-analysis with *Ranger* requires as input either 1) a text file already containing the sample-wise metrics for each task (in the GLUE example) or 2) a text file containing the retrieval runs and the qrels containing the labels (in the BEIR example).

The paths to the text files for each task are stored in a config.yaml file and read in with the class `ClassificationLocationConfig` or `RetrievalLocationConfig`. The entry point for loading the data and possibly computing metrics is `load_and_compute_metrics(name, measure, config)`. Having the treatment and control data, we can analyze the effects and compute effect sizes:

```

from ranger.metric_containers import
AggregatedPairedMetrics, AggregatedMetrics
from ranger.meta_analysis import
analyze_effects

effects = AggregatedPairedMetrics(
    treatment=t.get_metrics(),
    control=c.get_metrics())
eff_size = analyze_effects(
    list(conf.display_names.values()),
    effects=effects,
    effect_type="SMD")

```

Here the `effect_type` variable refers to the type of difference measurement in the meta-analysis as

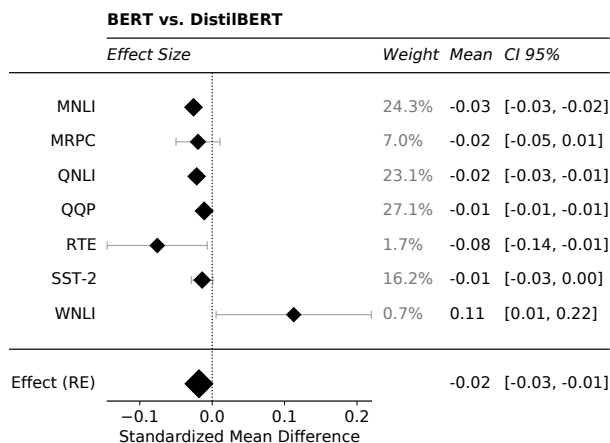


Figure 2: Forest plot of *Ranger* toolkit for tasks of the GLUE benchmark. Comparison in terms of accuracy between BERT and DistilBERT.

introduced in the previous section. The choice is between Raw Mean differences ("MD"), standardized mean differences ("SMD") or correlation ("CORR"). In order to visualize the effects, *Ranger* produces beautiful forest plots:

```
from ranger.forest_plots import forest_plot

plot = forest_plot(title=title,
experiment_names=list(config.display_names
.values()),
label_x_axis="Standardized Mean Difference",
effect_size=eff_size,
fig_width=8,
fig_height=8)
```

4 Case Study NLP: GLUE benchmark

In order to demonstrate the usage of the *Ranger* toolkit for various multi-task benchmarks, we conduct an evaluation on the popular General Language Understanding (GLUE) Benchmark (Wang et al., 2018).

We train and compare two classifiers on the GLUE benchmark: one classifier based on BERT (Devlin et al., 2018), the latter based on a smaller, more efficient transformer model trained on BERT scores, namely DistilBERT (Sanh et al., 2019)².

The official evaluation metric for two of the nine tasks (for CoLA and STS-B) is a correlation-based metric. Since these correlation-based metrics can not be computed sample-wise for each sample in the test set, the effect-size based meta-analysis can not be applied to those metrics and we exclude these two tasks from our evaluation.

²Checkpoints from Huggingface. bert-base-cased for BERT, distilbert-base-cased for DistilBERT.

We conduct the effect-size based meta-analysis based on the accuracy as metric and use Standardized Mean Difference to measure the effect-size (type in *Ranger* toolkit is 'SMD'). We illustrate the meta-analysis of the BERT and DistilBERT classifier in Figure 2. We also publish the Walk-you-Through Jupyter notebook in the *Ranger* toolkit to attain this forest plot for GLUE.

The location of the black diamonds visualizes the effect of the treatment (DistilBERT) compared to the baseline (BERT), whereas the size of the diamonds refers to the weight of this effect in the overall summary effect. We can see that using DistilBERT as base model for the classifier compared to BERT, has overall effect of a minor decrease in effectiveness. This behaviour is similar with the results on the MNLI, QNLI, and QQP where we also notice that the confidence intervals are very narrow or even non existent in the forest plot. For MRPC and SST-2 there is also a negative effect, however the effect is not significant, since the confidence intervals overlap with the baseline performance. For RTE and WNLI the effect of using DistilBERT compared to BERT is rather big compared to the summary effect, where for RTE the mean is 8% lower and for WNLI the mean is 11% higher than for the BERT classifier. However the large confidence intervals of these tasks indicate the large variability in the effect and thus the weight for taking these effects into account in the summary effect are rather low (0.7% and 1.7%).

Overall the summary effect shows that the DistilBERT classifier decreases effectiveness consistently by 2%. Since the confidence intervals are so narrow for the overall effect and do not overlap with the baseline (BERT classifier), we see that the overall effect is also significant.

5 Case Study IR: BEIR benchmark

Especially in IR evaluation, where it is common to evaluate multiple tasks with metrics, which are not comparable over different tasks (Thakur et al., 2021), we see a great benefit of using *Ranger* to aggregate the results of multiple tasks into one comparable statistical analysis. Thus we demonstrate the case study of using the *Ranger* toolkit for evaluation on commonly used IR collections, including a subset of the BEIR benchmark (Thakur et al., 2021). We presented this study originally as part of (Hofstätter et al., 2022), and the *Ranger* toolkit is a direct descendent of these initial experiments.

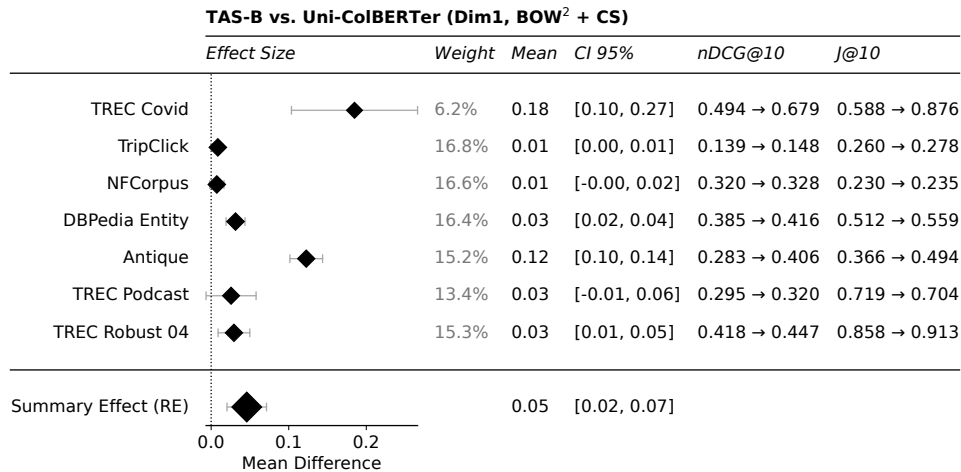


Figure 3: Forest plot of *Ranger* toolkit for tasks of the BEIR benchmark. Comparison in terms of nDCG@10 between TAS-B and ColBERTer.

We select tasks, which either 1) were annotated at a TREC³ track and thus contain high quality judgements, 2) were annotated according to the Cranfield paradigm (Cleverdon, 1967) or 3) contain a large amount of labels. All collections are evaluated with the ranking metric nDCG@10. We compare zero-shot retrieval with TAS-B (Hofstätter et al., 2021) as baseline to retrieval with Uni-ColBERTer (Hofstätter et al., 2022) as treatment.

We conduct a meta-analysis of the evaluation results based on nDCG@10 as metric and measure the effect-size with the mean difference (type is 'MD'). The output of the *Ranger* toolkit is illustrated in Figure 3. We publish a walk-through Jupyter notebook in the *Ranger* toolkit to attain this forest plot for BEIR benchmark evaluation.

In Figure 3 the effect size, the weight of the effect on the overall effect as well as the mean and confidence intervals of the effect are visualized. As an extension for IR we also visualize the nDCG@10 performance and J@10 judgement ratio from baseline → to treatment.

For NFCorpus and TREC Podcast we see a small positive effect of Uni-ColBERTer compared to TAS-B, however the confidence intervals are overlapping with the baseline performance indicating no clear positive effect on these tasks. For TripClick, DBPedia Entity and TREC Robust 04 we see a consistent and significant small positive effect with narrow confidence intervals of Uni-ColBERTer and this effect is even greater for Antique and TREC Covid. Notice the great confidence intervals for TREC Covid, since the evaluation of TREC Covid is only based on 50 queries and thus

its influence for the overall effect should be and is the lowest (6.2%) among the test sets.

The judgement ratio J@10 in the left most column shows the percentage of judged documents in the Top 10 of retrieved results. Analyzing the judgement ratio one can also get an understanding of how reliable the evaluation results are and how comparable the results of the two different retrieval models are, since a high difference in judgement ratio could indicate lower comparability of the two models with the respective test set.

Overall the summary effect of Uni-ColBERTer compared to TAS-B is consistent and significantly positive, increasing effectiveness by 0.05.

6 Conclusion

We presented *Ranger* – a task-agnostic toolkit for easy-to-use meta-analysis to evaluate multiple tasks. We described the theoretical basis on which we built our toolkit; the implementation and usage; and furthermore we provide two cases studies for common IR and NLP settings to highlight capabilities of *Ranger*. We do not claim to have all the answers, nor that using *Ranger* will solve all your multi-task evaluation problems. Nevertheless, we hope that *Ranger* is useful for the community to improve multi-task experimentation and make its evaluation more robust.

Acknowledgements This work is supported by the Christian Doppler Research Association (CDG) and has received funding from the EU Horizon 2020 ITN/ETN project on Domain Specific Systems for Information Extraction and Retrieval (H2020-EU.1.3.1., ID: 860721).

³<https://trec.nist.gov/>

References

- Leif Azzopardi, Paul Thomas, and Alistair Moffat. 2019. [Cwl_eval: An evaluation tool for information retrieval](#). In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19*, page 1321–1324, New York, NY, USA. Association for Computing Machinery.
- Michael Borenstein, Larry V Hedges, Julian PT Higgins, and Hannah R Rothstein. 2009. *Introduction to meta-analysis*. John Wiley & Sons, Ltd.
- Samuel Bowman. 2022. [The dangers of underclaiming: Reasons for caution when reporting how NLP systems fail](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7484–7499, Dublin, Ireland. Association for Computational Linguistics.
- Samuel R. Bowman and George Dahl. 2021. [What will it take to fix benchmarking in natural language understanding?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855, Online. Association for Computational Linguistics.
- Kenneth Church, Mark Liberman, and Valia Kordoni. 2021. [Benchmarking: Past, present and future](#). In *Proceedings of the 1st Workshop on Benchmarking: Past, Present and Future*, pages 1–7, Online. Association for Computational Linguistics.
- Cyril Cleverdon. 1967. *The Cranfield Tests on Index Language Devices*. San Francisco, CA, USA.
- Pierre Colombo, Nathan Noiry, Ekhine Irurozki, and Stephan Cl  men  on. 2022. [What are the best systems? new perspectives on nlp benchmarking](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 26915–26932. Curran Associates, Inc.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. [Overview of the trec 2021 deep learning track](#). In *Text REtrieval Conference (TREC)*. TREC.
- Rebecca DerSimonian and Nan Laird. 2015. [Meta-analysis in clinical trials revisited](#). *Contemporary Clinical Trials*, 45:139–145. 10th Anniversary Special Issue.
- 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*.
- Larry V Hedges. 1981. Distribution theory for glass’s estimator of effect size and related estimators. *Journal of Educational Statistics*, 6(2):107–128.
- Sebastian Hofst  tter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. [Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction](#).
- Sebastian Hofst  tter, 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 (SIGIR '21)*.
- Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. [Streamlining evaluation with ir-measures](#). In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*, page 305–310, Berlin, Heidelberg. Springer-Verlag.
- 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*.
- Ian Soboroff. 2018. Meta-analysis for retrieval experiments involving multiple test collections. In *Proc. of CIKM*.
- Nandan Thakur, Nils Reimers, Andreas R  ckl  , Abhishek Srivastava, and Iryna Gurevych. 2021. [Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models](#).
- Ellen M. Voorhees and Kirk Roberts. 2021. [On the quality of the TREC-COVID IR test collections](#). In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2422–2428. ACM.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

GAIA Search: Hugging Face and Pyserini Interoperability for NLP Training Data Exploration

Aleksandra Piktus^{1,2} Odunayo Ogundepo³ Christopher Akiki^{4,5} Akintunde Oladipo³
Xinyu Zhang³ Hailey Schoelkopf⁶ Stella Biderman^{6,7} Martin Potthast^{4,5} Jimmy Lin³

¹Hugging Face ²Sapienza University ³University of Waterloo
⁴Leipzig University ⁵ScaDS.AI ⁶EleutherAI
⁷Booz Allen Hamilton
piktus@huggingface.co

Abstract

Noticing the urgent need to provide tools for fast and user-friendly qualitative analysis of large-scale textual corpora of the modern NLP, we propose to turn to the mature and well-tested methods from the domain of Information Retrieval (IR)—a research field with a long history of tackling TB-scale document collections. We discuss how Pyserini—a widely used toolkit for reproducible IR research can be integrated with the Hugging Face ecosystem of open-source AI libraries and artifacts. We leverage the existing functionalities of both platforms while proposing novel features further facilitating their integration. Our goal is to give NLP researchers tools that will allow them to develop retrieval-based instrumentation for their data analytics needs with ease and agility. We include a Jupyter Notebook-based walk through the core interoperability features, available [on GitHub](#). We then demonstrate how the ideas we present can be operationalized to create a powerful tool for qualitative data analysis in NLP. We present GAIA Search—a search engine built following previously laid out principles, giving access to four popular large-scale text collections. GAIA serves a dual purpose of illustrating the potential of methodologies we discuss but also as a standalone qualitative analysis tool that can be leveraged by NLP researchers aiming to understand datasets prior to using them in training. GAIA is hosted live on [Hugging Face Spaces](#).

1 Introduction

Training large language models, or LLMs (Brown et al., 2020; Lieber et al., 2021; Rae et al., 2021; Smith et al., 2022; Le Scao et al., 2022; Chowdhery et al., 2022; Touvron et al., 2023), established itself as the central task of the modern Natural Language Processing (NLP) research. The attempts to understand the scaling laws of LLMs led researchers to believe that simply increasing the number of parameters may not bring the desired improvements without a simultaneous increase in the size of the

LLM training data (Kaplan et al., 2020; Hoffmann et al., 2022). These observations only increased an already pressing need for massive textual datasets, fueling the proliferation of Web-based corpora of TB-scale created with varying levels of curation and quality control.

Rather than investing in scraping the Web on their own, dataset creators typically turn to Common Crawl¹ as the main source of text to include in their corpora. A repository of Web snapshots dating back to 2011, Common Crawl contains various types of low-quality text (Luccioni and Viviano, 2021). Pre-processing steps commonly introduced by dataset creators aiming to filter out undesired content include removing any documents with words matching a pre-defined, static blacklist, like in the case of C4 (Raffel et al., 2020), perplexity-based filtering like in CCNet and ROOTS (Wenzek et al., 2019; Laurençon et al., 2022), removing malformed text via simple text statistics like in the case of OSCAR (Abadji et al., 2022) or through deduplication, studied extensively by Lee et al. (2022). However, the generated artifacts still tend to contain a multitude of worrying phenomena, such as synthetic data (Dodge et al., 2021), private and copyrighted data (Huang et al., 2022) or incorrect language codes and translations (Kreutzer et al., 2022). A lack of representation of diversity and socio-cultural and socio-economic biases constitute another big challenge of Common Crawl and datasets derived from it (Bender et al.; Blodgett et al., 2020; Field et al., 2021; Stanczak and Augenstein, 2021; Beaulieu and Leonelli, 2021).

Aware of the mounting problems with training data for modern LLMs, and appreciating the value of data exploration for better modeling in general, we focus our current work on building tools that can facilitate the qualitative analysis of NLP datasets. We propose to leverage the extensive experience of the Information Retrieval community in build-

¹<https://commoncrawl.org/>

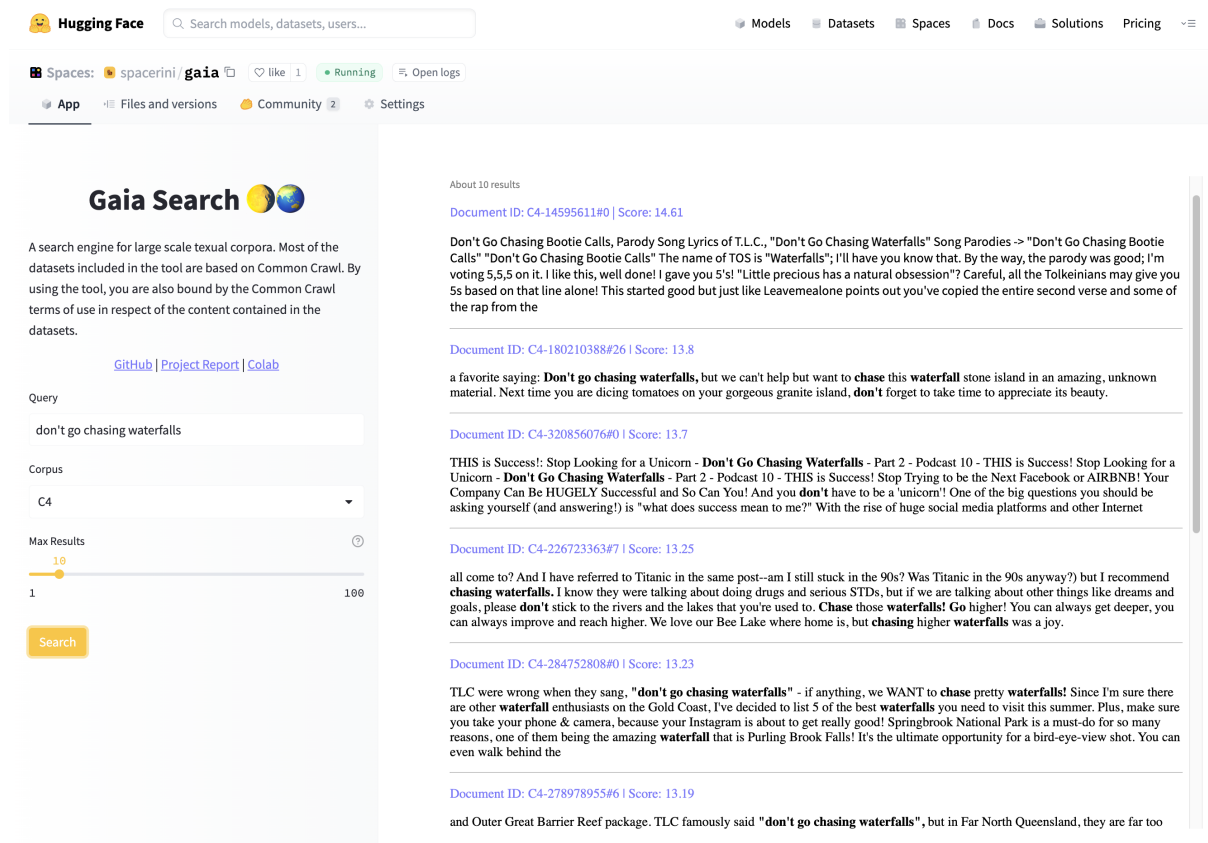


Figure 1: The user interface of GAIA Search.

ing relevance-based search indices for large-scale document collections and put it into practice in the context of NLP data exploration work. We follow with a demonstration of ways in which the interoperability between Pyserini (Lin et al., 2021), a leading toolkit for reproducible IR research on one side, and Hugging Face², a platform for open AI research on the other, can be leveraged to build tools for easy and effective analysis of textual data. To facilitate the adoption of the proposed methods we provide a collection of Jupyter Notebooks with step-by-step explanations of explored functionalities available at github.com/huggingface/gaia/tree/main/notebooks.

Finally, we release GAIA—a simple, yet powerful search engine giving relevance-based interface to four popular, large-scale, textual datasets, namely C4 (Raffel et al., 2020), the Pile (Gao et al., 2021; Biderman et al., 2022), ROOTS (Laurençon et al., 2022) and captions from LAION-2B-en (Schuhmann et al., 2022). All considered datasets rely to a big extent on data mined from Common Crawl. GAIA benefits from the interoperability between Pyserini and Hugging Face that

²<https://huggingface.co/>

we discuss in the first part of the paper, while also constituting a standalone contribution which can benefit the NLP research community by making it easy to study leading corpora qualitatively. GAIA is available online at hf.co/spaces/spacerini/gaia.

2 Background

The ability to analyze large collections of textual data is core in multiple research and engineering disciplines. While the industrial standard is to rely on robust, scalable database and data analytics infrastructure, in the research environment, we typically resort to more local, granular and flexible, if ad-hoc, solutions which leverage toolkits such as NumPy (Harris et al., 2020), Pandas (pandas development team, 2020; Wes McKinney, 2010), SciPy (Virtanen et al., 2020) and others. A common research approach to data analytics involves using one of the aforementioned packages in combination with Jupyter Notebooks³. Notebooks make it easy to deploy and share analyses, however, typically they remain essentially non-interactive, requiring

³<https://jupyter.org/>

Dataset	Reference	Hugging Face Hub link	# docs	# snippets	Data Size	Index Size
C4	Raffel et al. (2020)	c4	365M	1,587M	829GB	1.3TB
The Pile	Gao et al. (2021)	the_pile_deduplicated	134M	673M	825GB	1.2TB
ROOTS	Laurençon et al. (2022)	bigscience-data	598M	2,171M	1.6TB	2.6TB
LAION	Schuhmann et al. (2022)	laion2B-en	2,322M	1,351M	503GB	446GB
Total			3,419M	5,782M	3.76 TB	5.55TB

Table 1: Datasets included in the GAIA Search tool. All numbers refer to the size of the train split of the data.

at least a basic understanding of programming to be able to work with them efficiently. With the commodification of AI, and NLP in particular, and the expansion of NLP technologies into research areas beyond AI (Yang et al., 2022; Smith et al., 2015; Bhardwaj et al., 2017; Niezni et al., 2022), the need for easy to use, no-code tools for understanding AI artifacts arises. This need is partly addressed by Python packages such as Streamlit⁴ and Gradio⁵, designed to facilitate the creation of interactive Machine Learning (ML) demos. As the authors of the Gradio white paper (Abid et al., 2019) point out, the accessibility and ease of use of the analysis tools is critical if we want to build an understanding of AI and trust in it. The Hugging Face Spaces platform, providing free hosting of both Streamlit, Gradio, and Docker-based applications, serves this exact purpose. However, it puts emphasis on demonstrating the capabilities of models while paying less attention to the datasets used to train them.

Even more so than in NLP, the evaluation of IR systems is heavily dependent on the implementation details of the retrieval systems serving the search indices being evaluated. The lack of standardisation of IR evaluation was the main motivator behind creating Anserini (Yang et al., 2017), a Lucene⁶-based toolkit for reproducible IR research, and the follow-up Pyserini (Lin et al., 2021)—a convenient Python API to the underlying Java-based implementation of Anserini. While it is relatively easy to build and serve search indices backed by Pyserini and Lucene, the task of building and deploying interactive user interfaces generally comes with a higher engineering barrier of entry.

Relevance-based search interfaces have been previously explored in the context of NLP—e.g. in the C4 (Raffel et al., 2020) analysis (Dodge et al., 2021), in COVID-related datasets (Zhang et al., 2020) or in news quotes (Vuković et al., 2022).

⁴<https://streamlit.io/>

⁵<https://gradio.app/>

⁶<https://lucene.apache.org/>

Rather than focusing only on providing finished artifacts, however, we intend our current work to serve as a reference and inspiration for NLP researchers looking to develop and deploy similar applications by themselves.

We attempt to bring together the power of Pyserini-backed retrieval and the agility of ML demo development within the Hugging Face ecosystem to serve the goal of building intuitive data exploration tools. We believe that resulting applications will make a great difference for NLP researchers trying to study their data qualitatively, as well as to non-technical researchers looking for tools allowing them to perform dataset analysis in a no-code fashion. We propose our search engine GAIA as a compelling case in point.

3 Pyserini and Hugging Face: From Data to Search

In the current section we discuss core components which need to be considered when building a search application for textual datasets. We focus on how each step can be facilitated by the use of Pyserini, Hugging Face, or a combination of the two. We also provide hands-on tutorials covering basic concepts and search engine building blocks such as [data loading and indexing](#), [tokenization](#), [search](#), and [index analysis](#). We further release the [pre-processing](#), [backend](#) and [frontend code](#) that allowed us to index 3.5 billion documents—chunked into 5.8 billion snippets—and serve 5.55TB worth of BM25 indexes.

3.1 Data Access

The Hugging Face hub is the repository of over 20,000 datasets from across AI domains. This includes the most popular large-scale text corpora in NLP—for example all the datasets we consider in GAIA (see Table 1 for details), but also other popular large scale text datasets such as OS-CAR (Abadji et al., 2022) and The Stack (Kocetkov et al., 2022) among many others. Each dataset

hosted on the Hub can be accessed locally using the datasets (Lhoest et al., 2021) library which provides convenient and parallelizable APIs for downloading and processing the data. Memory-mapping is supported by default and uses the efficient an Apache Arrow format,⁷ making it possible to seamlessly handle datasets surpassing the RAM constraints of a given machine. Datasets also provide a streaming functionality which dispenses of downloading data to disk, making it possible to work with larger-than-disk datasets.

3.2 Tokenization

Tokenization is a crucial pre-processing step in NLP in general, and Information Retrieval in particular. In the context of IR, this process typically includes removing stop words, stemming, lemmatization, and removing non-alphanumeric characters. By default, Pyserini uses Lucene analyzers—heuristics-based algorithms designed for various languages and use cases, to tokenize text. The drawback of this approach is that only some languages have dedicated analyzers, while others have to resort to simply breaking on whitespace, which inadvertently leads to suboptimal performance.

An alternative to whitespace tokenization that has shown promise in Information Retrieval and is a mainstay in NLP is subword tokenization (Mielke et al., 2021), a process which splits words into smaller units based on their frequency in the corpus. Hugging Face provides a range of tokenizers that are specifically designed to work with its pre-trained transformer language models, as well as the means to train such tokenizers (MOI et al., 2022).

As of recently, Pyserini can leverage Hugging Face pre-trained subword tokenizers to improve indexing and searching for multiple languages. Pre-trained tokenizers from Hugging Face can serve as drop-in replacements for Lucene Analyzers, improving retrieval effectiveness, particularly in low-resource languages (Ogundepo et al., 2022). This interoperability between Hugging Face and Pyserini makes it easy for researchers to incorporate deep learning-based language models into their information retrieval workflows and opens up new avenues for research in the field.

3.3 Building the Index

Indexing constitutes the core functionality of Pyserini. The library enables experiments with bag-of-

⁷<https://arrow.apache.org/>

words sparse retrieval using Lucene, and dense vector retrieval using Faiss (Johnson et al., 2019), as well as hybrid retrieval combining the two. Though this project focuses solely on sparse retrieval using BM25 indexes, Pyserini’s dense encoding and retrieval API would make it very easy to adapt all examples and demos to this paradigm.

Offline Indexing. Arrow-backed Hugging Face datasets readily lend themselves to being indexed by Pyserini’s standard Lucene indexer. In principle, one can build an index of a Hugging Face dataset simply by downloading it locally and then passing the file path to the Pyserini indexer via a command line argument. The scenario where a pre-processing step is required in between the data download and the indexing step—as with document segmentation which we discuss later in Section 4—can be realised straightforwardly for smaller datasets, which fit both on disk and into RAM. The larger-than-RAM datasets which fit on disk, can be easily sharded into any of the disk text formats supported by Pyserini (those include CSV, TSV, JSON, and JSONL) and processed concurrently within RAM limits to be then passed to the indexer.

Datasets Streaming. As of recently, it is also possible to index datasets which don’t fit on disk.⁸ This new addition to Pyserini—one that resulted out of our current collaboration—allows users to stream text into the index directly—in other words, build an index on the fly from a text stream rather than from a static file saved on disk. As a result, larger-than-disk collections can be streamed from the Hugging Face Hub directly into the local indexing process. Data streaming can also improve experimental agility for smaller datasets, by removing the data downloads step from the Hugging Face dataset—Pyserini index pipeline.

3.4 Backend: Custom Pyserini Server

Once the data index is ready we need a way to host it and serve the search functionality to the clients. We propose a simple Python-based, Pyserini server implementation for GAIA, which can be easily generalized to other use-cases. The server code can be accessed [on GitHub](#).

⁸Note however, that the resulting index does have to fit on disk. As a result, we envision this functionality to be particularly convenient for scenarios where either the dataset or the index may be able to fit on disk, but both do not—a common scenario when dealing with TB-scale artefacts.

3.5 Frontend: Interactive Demos

Providing interactive demos which enable the exploration of AI artifacts is crucial in order to be able to collaborate across research disciplines and share results with colleagues without imposing the burden of setting up their own engineering stack on them. By offering the hosting of Gradio and Streamlit applications Hugging Face Spaces meet this need perfectly. We encourage readers to follow the implementations of GAIA for an example of how to build a simple UI for a search tool.

4 Case Study: GAIA Search

Relevance-based search tools have the potential of the largest impact on massive-scale datasets, common in modern NLP. Unlike with smaller data collections, where simpler investigation strategies, e.g. via a combination of Pandas and Jupyter Notebooks, may be feasible, huge datasets are generally too cumbersome to process this way. A big benefit of search engines in the form that we propose is also the fact that after being set up, they require no engineering skills or extensive computing resources to operate, expanding the community of potential users. We demonstrate this with GAIA search, available online at hf.co/spaces/spacerini/gaia.

4.1 Included Datasets

GAIA proposes a simple interface to four large-scale textual datasets—C4, The Pile, ROOTS, and captions from LAION-2B-en. The reader may consult Table 1 for details on respective datasets. All of the datasets included in GAIA are sourced at least partly from Common Crawl. The users of the tool are therefore bound by the Common Crawl terms of use⁹ in respect of the content contained in the datasets. Additionally, in order to respect the data subjects’ rights (Jernite et al., 2022) we refrain from presenting full documents in the tool, and instead include snippets of at most 256 words. We redact the personally identifiable information (PII) on all search results on the backend side, using the PII redaction script open-sourced alongside the BigScience¹⁰ language model BLOOM (Le Scao et al., 2022). Below we discuss details of the respective datasets’ pre-processing.

⁹<https://commoncrawl.org/terms-of-use/>

¹⁰bigscience.huggingface.co

C4. This is a dataset fully sourced from Common Crawl. We index the variant of the English split of the dataset available on the Hugging Face hub. C4 has been used to train T5 (Raffel et al., 2020), a major seq-2-seq model with a plethora of downstream applications, parts of it have also contributed to the training of other LLMs, e.g. LaMDA (Thoppilan et al., 2022) and Chinchilla (Hoffmann et al., 2022), which makes it a compelling dataset to study.

The Pile. This corpus has been a standard dataset for many English LLM releases from various organizations (Biderman et al., 2023; Black et al., 2021; Wang and Komatsuzaki, 2021; Black et al., 2022; Smith et al., 2022; Tang, 2021; Zhang et al., 2022; Lieber et al., 2021), so we believe that it is important to expose its contents to public view. The Pile is an English-only corpus containing multiple sub-corpora from various sources (Biderman et al., 2022). We use a variant of The Pile which has been deduplicated with MinhashLSH and a threshold of 0.87, following the advice of Lee et al. (2022). Notably, this variant of the Pile has also been used to train an LLMs (Biderman et al., 2023). We hope that providing the search interface will allow further investigation of the subjective differences between deduplicated and unprocessed corpora. Both the canonical variant of The Pile and its deduplicated counterpart are available on the Hugging Face Hub.

ROOTS. Developed for the purpose of training BLOOM (Le Scao et al., 2022), this is the only multilingual dataset available in GAIA. We therefore, create independent indices for each language or language group provided in the corpus, resulting in 13 separate indices—Arabic, Catalan, Code (comprising all programming languages included in the corpus), English, Spanish, Basque, French, Indonesian, Indic and Niger-Congo (language groups), Portuguese, Vietnamese and Chinese. We return results for each index when issuing queries in the tool.

LAION-2B-en LAION is a dataset of [image caption, image URL] pairs scraped from the Web. It has been used to train Stable Diffusion (Rombach et al., 2021), a textual-prompt-based image generation model, constituting an open-source counterpart to OpenAI’s DALL·E 2 (Ramesh et al., 2022). We use LAION-2B-en, the subset of the original dataset with captions in English, as the starting point for further pre-processing. We start by dedu-

plicating captions, which yields clusters of image URLs with identical captions (deduplication code is available [on GitHub](#)). We then index unique captions. For textual queries to our tool, we return results consisting of the relevant captions. Alongside each result, we include the list of associated image URLs.

4.2 Implementation and Functionality

The implementation of GAIA makes use of a variety of interoperability features we've discussed in Section 3. As detailed in Table 1, all of the considered datasets are available on the Hugging Face Hub. We download and segment them locally. Such segmented datasets are then provided as input to a Pyserini indexer. We leverage Streamlit to build the user interface for our tool and host it on Hugging Face Spaces. On the backend side, the indices are served from Hugging Face provisioned machines. We open-source helper functions for segmenting long documents and the backend server code at github.com/huggingface/gaia.

5 Limitations and Future Plans

A major area for consideration when developing data access tools is that of data governance, privacy and data ownership (Jernite et al., 2022; Carlini et al., 2020). In our current work we focus on the technical aspects of giving access to large data collections, however, we urge users to consider data governance principles when designing their own tools. In terms of the infrastructure, the cost and complexity of hosting the retrieval index falls on the creator of the tool, which can be easy to manage for small datasets but becomes more problematic when entering the realm of TB-scale corpora. We are currently investigating a parallel workstream that could address this limitation at least partly.

6 Conclusions

We showcase interoperability between Hugging Face and Pyserini and provide value to the NLP community by demonstrating easy ways to perform high-quality, large-scale retrieval with open-source tools. We also introduce GAIA - a search engine for retrieval-based exploration of four major textual datasets. We wish to encourage NLP and IR practitioners to follow our examples and build their own tools to explore both large and smaller-scale textual datasets.

7 Acknowledgements

Authors would like to thank Carlos Muñoz Ferlandis, Daniel van Strien, Katie Link and Quentin Lhoest for valuable tips and suggestions. This research was also supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

8 Impact Statement

As mentioned in Section 5, accessing large-scale, web-scraped textual corpora comes with a variety of ethical considerations, pertaining to the protection of rights of the data owners and people whose privacy or copyright might be infringed upon. We introduce guardrails, namely the PII redaction and the segmentation of documents into short snippets, preventing the ability to reconstruct full documents or full corpora, into the GAIA Search design. We strongly encourage researchers aiming to build similar tools to do the same. Overall, a lot of these problems seem to occur because we're proposing the tool only after the datasets have been created and models trained on them. The workflow we envision for future research projects would involve building data exploration tools prior to the release of the datasets, so that core problems can be observed, studied and addressed before datasets reach an external audience.

References

- Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. [Towards a cleaner document-oriented multilingual crawled corpus](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.
- Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. [Gradio: Hassle-free sharing and testing of ml models in the wild](#).
- Anne Beaulieu and Sabina Leonelli. 2021. *Data and Society: A Critical Introduction*. Sage.
- Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big?, year = 2021, isbn = 9781450383097, publisher = Association for Computing Machinery, address = New York, NY, USA, url = <https://doi.org/10.1145/3442188.3445922>, doi = 10.1145/3442188.3445922, abstract = The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language

- models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models., booktitle = Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pages = 610–623, numpages = 14, location = Virtual Event, Canada, series = FAccT '21.
- Rohan Bhardwaj, Ankita R. Nambiar, and Debojyoti Dutta. 2017. [A study of machine learning in health-care](#). In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 236–241.
- Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. [Datasheet for the pile](#). *CoRR*, abs/2201.07311.
- Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. [Pythia: a scaling suite for language model interpretability research](#). *Computing Research Repository*. Version 1.
- Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. [GPT-Neo: Large scale autoregressive language modeling with Mesh-TensorFlow](#). *GitHub*.
- Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.
- Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. [Language \(Technology\) is Power: A Critical Survey of “Bias” in NLP](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, 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, 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.
- Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. [Extracting Training Data from Large Language Models](#). *arXiv:2012.07805 [cs]*.
- 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](#).
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. [Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Anjalie Field, Su Lin Blodgett, Zeerak Waseem, and Yulia Tsvetkov. 2021. [A Survey of Race, Racism, and Anti-Racism in NLP](#). 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 1905–1925, Online. Association for Computational Linguistics.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang,

- Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. [The pile: An 800gb dataset of diverse text for language modeling](#).
- Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. [Array programming with NumPy](#). *Nature*, 585(7825):357–362.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. [Training compute-optimal large language models](#).
- Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. [Are Large Pre-Trained Language Models Leaking Your Personal Information?](#)
- Yacine Jernite, Huu Nguyen, Stella Biderman, Anna Rogers, Maraim Masoud, Valentin Danchev, Samson Tan, Alexandra Sasha Luccioni, Nishant Subramani, Isaac Johnson, Gerard Dupont, Jesse Dodge, Kyle Lo, Zeerak Talat, Dragomir Radev, Aaron Gokaslan, Somaieh Nikpoor, Peter Henderson, Rishi Bommasani, and Margaret Mitchell. 2022. [Data governance in the age of large-scale data-driven language technology](#). In *2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22*, page 2206–2222, New York, NY, USA. Association for Computing Machinery.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#).
- Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. [The stack: 3 tb of permissively licensed source code](#).
- Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroko Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhlov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. [Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets](#). *Transactions of the Association for Computational Linguistics*, 10:50–72.
- Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Froberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Romero Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Vu Minh Chien, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Luccioni, and Yacine Jernite. 2022. [The bigscience ROOTS corpus: A 1.6TB composite multilingual dataset](#). In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovich, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios,

Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Froberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nuru-laqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, So-maieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M. Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanjit Gandhi, Shaden Smith, Stéphane Requery, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névél, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Nae-joung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruo Chen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigoná Unldreaj, Arash Aghagol, Are-

zoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Onon-iwu, Habib Rezanejad, Hessie Jones, Indrani Bhat-tacharya, Irene Solaiman, Irina Sedenko, Isar Ne-jadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim El-badri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Ra-jani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Al-izadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguié, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrmann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A. Castillo, Mari-anna Nezhurina, Mario Sängler, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myung-sun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S. Deshmukh, Shubhanshu Mishra, Sid Ki-blawi, Simon Ott, Sinee Sang-aoonsiri, Srishti Ku-mar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022. [BLOOM: A 176B-Parameter Open-Access Multilingual Language Model](#). In *Thirty-Sixth Conference on Neural Information Processing Systems*, New Orleans, Louisiana. arXiv.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. [Deduplicating training data makes language models better](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis,

- Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. [Datasets: A community library for natural language processing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. Technical report, AI21 Labs.
- Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.
- Alexandra Luccioni and Joseph Viviano. 2021. [What’s in the box? an analysis of undesirable content in the Common Crawl corpus](#). 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 182–189, Online. Association for Computational Linguistics.
- Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *ArXiv*, abs/2112.10508.
- Anthony MOI, Nicolas Patry, Pierric Cistac, Pete, Fun-towicz Morgan, Sebastian Pütz, Mishig, Bjarte Johansen, Thomas Wolf, Sylvain Gugger, Clement, Julien Chaumond, Lysandre Debut, François Garillot, Luc Georges, dcelus, JC Louis, MarcusGrass, Taufiqzaman Peyash, 0xflotus, Alan deLevie, Alexander Mamaev, Arthur, Cameron, Colin Clement, Dagmawi Moges, David Hewitt, Denis Zolotukhin, and Geoffrey Thomas. 2022. [huggingface/tokenizers: Rust 0.13.2](#).
- Danna Niezni, Hillel Taub-Tabib, Yuval Harris, Hagit Sason-Bauer, Yakir Amrusi, Dana Azagury, Maytal Avrashami, Shaked Launer-Wachs, Jon Borchardt, M Kusold, Aryeh Tiktinsky, Tom Hope, Yoav Goldberg, and Yosi Shamay. 2022. [Extending the boundaries of cancer therapeutic complexity with literature data mining](#). *bioRxiv*.
- Ogunayo Ogundepo, Xinyu Zhang, and Jimmy Lin. 2022. [Better than whitespace: Information retrieval for languages without custom tokenizers](#).
- The pandas development team. 2020. [pandas-dev/pandas: Pandas](#).
- Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Mari-beth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsim-poukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d’Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. [Hierarchical text-conditional image generation with clip latents](#).
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. [High-resolution image synthesis with latent diffusion models](#).
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. [Laion-5b: An open large-scale dataset for training next generation image-text models](#).
- David A. Smith, Ryan Cordell, and Abby Mullen. 2015. [Computational Methods for Uncovering Reprinted Texts in Antebellum Newspapers](#). *American Literary History*, 27(3):E1–E15.
- Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George

- Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. [Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model.](#)
- Karolina Stanczak and Isabelle Augenstein. 2021. [A Survey on Gender Bias in Natural Language Processing.](#)
- Jie Tang. 2021. WuDao: Pretrain the world. Keynote address at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.
- Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguerre-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. [Lamda: Language models for dialog applications.](#)
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models.](#)
- Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. [SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.](#) *Nature Methods*, 17:261–272.
- Vuk Vuković, Akhil Arora, Huan-Cheng Chang, Andreas Spitz, and Robert West. 2022. [Quote erat demonstrandum: A web interface for exploring the quotebank corpus.](#) In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.
- Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model.
- Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. [Ccnnet: Extracting high quality monolingual datasets from web crawl data.](#)
- Wes McKinney. 2010. [Data Structures for Statistical Computing in Python.](#) In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.
- Peilin Yang, Hui Fang, and Jimmy Lin. 2017. [Anserini: Enabling the use of lucene for information retrieval research.](#) In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17*, page 1253–1256, New York, NY, USA. Association for Computing Machinery.
- Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, Christopher A Harle, Gloria Lipori, Duane A Mitchell, William R Hogan, Elizabeth A Shenkman, Jiang Bian, and Yonghui Wu. 2022. [Gatortron: A large clinical language model to unlock patient information from unstructured electronic health records.](#)
- Edwin Zhang, Nikhil Gupta, Raphael Tang, Xiao Han, Ronak Pradeep, Kuang Lu, Yue Zhang, Rodrigo Nogueira, Kyunghyun Cho, Hui Fang, and Jimmy Lin. 2020. [Covidex: Neural ranking models and keyword search infrastructure for the COVID-19 open research dataset.](#) In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 31–41, Online. Association for Computational Linguistics.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. [Opt: Open pre-trained transformer language models.](#)

DeepPavlov Dream: Platform for Building Generative AI Assistants

Dilyara Zharikova¹
dilyara.rimovna@gmail.com

Daniel Kornev¹
daniel@kornevs.org

Fedor Ignatov¹
ignatov.fedor@gmail.com

Maxim Talimanchuk¹

Dmitry Evseev¹

Kseniia Petukhova¹

Veronika Smilga¹

Dmitry Karpov¹

Yana Shishkina¹

Dmitry Kosenko¹

Mikhail Burtsev²
mbur@lims.ac.uk

¹Moscow Institute of Physics and Technology, Russia

²London Institute for Mathematical Sciences, UK

Abstract

An open-source DeepPavlov Dream Platform is specifically tailored for development of complex dialog systems like Generative AI Assistants. The stack prioritizes efficiency, modularity, scalability, and extensibility with the goal to make it easier to develop complex dialog systems from scratch. It supports modular approach to implementation of conversational agents enabling their development through the choice of NLP components and conversational skills from a rich library organized into the distributions of ready-for-use multi-skill AI assistant systems. In DeepPavlov Dream, multi-skill Generative AI Assistant consists of NLP components that extract features from user utterances, conversational skills that generate or retrieve a response, skill and response selectors that facilitate choice of relevant skills and the best response, as well as a conversational orchestrator that enables creation of multi-skill Generative AI Assistants scalable up to industrial grade AI assistants. The platform allows to integrate large language models into dialog pipeline, customize with prompt engineering, handle multiple prompts during the same dialog session and create simple multimodal assistants.

1 Introduction

Complex AI assistants are becoming more and more widespread. As a result, interest in technology for building complex conversational interfaces has grown significantly over the last years. At the same time, most of the available systems enabling complex dialog systems development are proprietary or limited. This opens up new opportunities for open-source systems that facilitate the development of complex AI assistants.

The DeepPavlov Dream Platform¹ provides a stack of Apache 2.0-licensed open-source technologies that enable development of complex dialog systems such as enterprise AI assistants. The platform features a conversational AI orchestrator called DeepPavlov Agent to coordinate an asynchronous scalable dialog pipeline; a framework called DeepPavlov Dialog Flow Framework (DFF) to facilitate development of the multi-step skills; support for Wikidata and custom knowledge graphs; a library of modern NLP components and conversational AI skills (script-based, chit-chat, question answering (QA), and generative skills) organized into a set of distributions of multi-skill conversational AI systems; and a visual designer. These components make it possible for developers and researchers to implement complex dialog systems ranging from multi-domain task-oriented or chit-chat chatbots to voice and multimodal AI assistants suitable for academic and enterprise use cases.

The DeepPavlov Dream Platform supports integrating large language models (LLMs) into production-ready dialog systems with the help of general and custom knowledge graphs (KGs) for fact checking, pre- and post-filters for filtering out unsuitable responses, and prompt-based generation for indirect control of LLMs.

Multimodality in DeepPavlov Dream provides an opportunity to operate with images and to perform actions via APIs based on the user's commands extracted during the conversation.

In this paper, we present the DeepPavlov Dream Platform for dialog systems development. In Section 2, we compare the platform with existing competitors. Sections 3, 4 and 5 introduce the pipeline's

¹<https://github.com/deeppavlov/dream>

orchestrator, distributions and other development tools. The platform’s components and proposed approaches for dialog system development are described in Section 6. Section 7 presents our methodology for building prompt-based Generative AI Assistants. Approaches to managing multimodality are described in Section 8. Finally, Section 9 provides an overview of a custom dialog system development process.

2 Comparison with Competitors

In Table 1 we present a comparison of the popular Conversational AI Platforms. Unlike other solutions, we provide an open-source Apache 2.0-based, multi-skill platform that enables development of complex open-domain dialog systems. DeepPavlov Dream allows for combining different response generation methods, adding pre- and post-filters, utilizing Wikidata and custom knowledge graphs, designing custom dialog management algorithms, integrating large language models (LLMs) to production-ready dialog systems. DeepPavlov Dream also provides simple integration with load-balancing tools that is crucial for LLMs-based dialog systems in production. We are also working towards text-based and multimodal experiences like robotics.

3 Pipeline

The DeepPavlov Dream is built upon the DeepPavlov Agent, an open-source framework for orchestrating complex systems. The full dialog system pipeline of the DeepPavlov Dream is presented in Figure 1. There are four component groups — Annotators, Skills, Candidate Annotators, and Response Annotators — and two dialog management components — Skill Selector and Response Selector. Dialog State is a shared memory that contains all the information about the dialog as DeepPavlov Agent expects that services are stateless and can be run as multiple instances. There are two synchronization points — Skill Selector and Response Selector. The other services can be run in parallel, although the Agent allows dependencies between services within the group to effectively chain them.

DeepPavlov Agent’s pipeline is asynchronous, i.e., one user’s request does not block the agent and does not prevent the agent from receiving and processing requests from other users. Each service is deployed in a separate docker container. In addition, separate containers are used for the agent itself

and the database. For development, one can run the dialog system locally using docker-compose or Kubernetes.

4 Distributions

Original DREAM Socialbot (Kuratov et al., 2020; Baymurzina et al., 2021b) included a large number of components for Natural Language Understanding (NLU) and Natural Language Generation (NLG). Different components are run independently and accept dialog state in a required format. Although some components may depend on the other ones’ annotations, in general modular system’s elements can be safely removed or replaced with their analogues. Platform users can re-use existing components to design their own dialog systems or develop custom components and include them into existing or custom dialog systems.

The DeepPavlov Dream Platform utilizes a distribution-based approach for dialog systems development. A *distribution* is a set of YAML-files specifying parameters of docker containers, and a configuration JSON-file determining a processing pipeline for DeepPavlov Agent. Platform contains different distributions including script-based English distributions, generative-based English, Russian and multi-lingual distributions, multimodal distribution, robot controller distribution, and lots of multi-skill distributions utilizing prompt-based generation with LLMs (details in Section 6).

5 Development Tools

Platform is supported by development tools — DeepPavlov dreamtools and DF Designer (Kuznetsov et al., 2021), a visual aid in developing scenario-driven skills for the Dream Platform using DFF².

DeepPavlov dreamtools is a set of tools in Python with a built-in command line tool which allows developers to operate with Platform distributions in a programmatic way.

Python Package The package exposes Dream distribution API via configuration and component objects. It supports a strict set of configuration files: one pipeline JSON-file and 4 docker-compose YAML configuration files (for production, development, proxy, and local deployments). For each file, the package implements configuration definitions

²https://github.com/deeppavlov/dialog_flow_framework

	DeepPavlov Dream	Mycroft AI	Linto AI	RASA	Amazon Lex	Google DialogFlow	IBM Watson	Avaamo	Kore.AI	Amelia
License	Apache 2.0	Apache 2.0	AGPL-3.0	Apache 2.0	N/A	N/A	N/A	N/A	N/A	N/A
Open Source	Yes	Yes	Yes	Yes	No	No	No	No	No	No
On-Premises	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes
Multi-skill	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes
Generative AI	Yes	Limited	Limited	Limited	No	Limited	Yes	Yes	Yes	Limited
FAQ Skills	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Task-oriented Skills	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Chit-Chat Skills	Yes	Limited	Limited	Limited	Limited	Limited	Limited	Limited	Limited	Yes
QA Skills	Wikidata, Custom	No	No	Custom	No	Custom	Custom	Custom	Custom	Custom
Knowledge Graphs	Wikidata, Custom KGs	No	No	Yes	No	Custom	Custom	Yes	Custom	Yes
Multilingual NLU	Per-lang, multilingual	Per-lang	Per-lang	Per-lang	Per-lang	Per-lang	Per-lang	Per-lang	Per-lang	Per-lang
Domain-specific NLU	No	No	Yes	3rd-party	Limited	3rd-party	Yes	Yes	Yes	Yes
External NLU	Yes	Yes	No	Yes	No	Partial	No	Yes	No	Unknown
Scalability	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multimodality	Text, Voice-3, Image(wip), Video(wip)	Text, Voice-3	Text, Voice	Text, Voice-3	Text, Voice	Text, Voice	Text, Voice	Text, Voice-3	Text, Voice-3	Text, Voice

Table 1: Comparison of popular Conversational AI Platforms: DeepPavlov Dream (DeepPavlov, 2023), Mycroft AI (AI, 2022b), Linto AI (AI, 2022a), RASA (RASA, 2022), Amazon Lex (Amazon, 2022), DialogFlow (Google, 2022), IBM Watson (Watson, 2022), Avaamo (Avaamo, 2022), Kore.AI (Kore.ai, 2022), Amelia (Amelia, 2022). In Multimodality row Voice-3 denotes usage of third-party applications for Voice processing



Figure 1: The high-level architecture of the user utterance’s processing in DeepPavlov Dream pipeline. The DeepPavlov Dream supports any number of components limited only by the available computational resources

as objects with methods for serializing, listing, creating, editing, and validating components. The components are defined as generic structures.

Additionally, its higher-level functionality allows for creating new distributions from scratch

or using the existing ones as a template, editing distributions, verifying consistency between component definitions (e.g., correct port forwarding, component naming, etc.), and creating DFF-based skills with all the necessary template files. We have

also transferred distribution testing utilities previously bundled with Dream: an automated dialog tester and a file validation script.

CLI DeepPavlov `dreamtools` can be used as a command line utility that wraps most of high-level package functionality. One of the common uses is creating a `local.yml` configuration file, which contains instructions on whether to deploy a component locally or redirect to a proxied DeepPavlov deployment.

6 Components

The platform contains English, Russian, and multilingual annotators and skills. Skill and Response Selectors are language-agnostic in general. The Skill Selector’s output may include skills not present in the running pipeline, making the Skill Selector’s code reusable. There are several available algorithms for Response Selector – tag-based for script-based distributions, LLM-based and ranking-based for generative distributions.

Modular container-based architecture of the Dream platform allows for integrating components from different frameworks and with different requirements. For many conversational NLP models, DeepPavlov (Burtsev et al., 2018) library is extensively used in several annotators and skills. DeepPavlov library, an open-source conversational NLP framework that is based on PyTorch and supports huggingface transformers, which allows it to use various transformer-based models from huggingface Hub (Burtsev et al., 2018). Traditionally, DeepPavlov contains several variants of models: the best-performing one with the highest score, the resource-efficient one with the fastest inference time (Kolesnikova et al., 2022), and the multilingual one with the support of several languages. DeepPavlov library provides ready-for-use tools for training and inference of NLP pipelines along with appropriate docker images that simplify integrating and adapting new components.

Annotators solve a variety of NLP tasks: text re-writing (sentence segmentation, punctuation recovery, spelling preprocessing, coreference resolution (Le et al., 2019)), text classification (sentiment, toxicity, emotions, factoidness, topics (Sagyndyk et al., 2022; Karpov and Burtsev, 2023)), dialog acts (Yu and Yu, 2019), intents and speech functions (Ostyakova et al., 2022)), token classification (NER, entity detection), knowledge re-

trieval (requests to external APIs, entity linking to KBs (Evseev), knowledge extraction from structured and textual KBs), text ranking (selecting most relevant prompts for conditional generation, response candidate ranking (Gao et al., 2020)).

Conversational input usually does not imply correct case sensitivity, so DeepPavlov Dream utilizes NER model (Chizhikova et al., 2023) adapted to both cased and uncased inputs. Additionally, the model is based on the Multilingual BERT (MBERT), which allows to support entity extraction in multiple languages due to the MBERT cross-lingual transferability (Kononov et al., 2020).

Resource consumption is also one of the most challenging parts of the dialog systems development. Often, the production-level annotators quality can be reached only by NN-based models each of which may require gigabytes of GPU memory. To tackle this problem, DeepPavlov equips Dream with a multi-task learning (MTL) classifier (Karpov and Kononov, 2023) trained to solve nine problems within a single model and decreasing GPU memory usage ninefold.

Skills in the Dream Platform define response generators. There are different types of algorithms for response generation, e.g., template-based, retrieval, and generative models. The skills that plan the dialog more than one step ahead are called scripted skills. These skills are able to get the dialog to develop in depth, which is already a generally accepted expectation of users from conversational systems. The scripts may utilize either slot-filling (Baymurzina et al., 2021a) in template-based responses or controllable generation via LLMs. Creating script-based skills by hands is a labour-consuming task, so we also researched approaches for automatic scripts generation (Kapelyushnik et al., 2022; Evseev et al.). A dialog system’s behavior may need to be deterministic in some cases. For that, developers may utilize either an intent-based templated response skill, a FAQ skill, or prompt-based generation via LLMs.

Prompt-based response generation via LLMs, a recent trend in NLP field (Bai et al., 2022; Taylor et al., 2022; Scao et al., 2022; Biderman et al., 2023; Köpf et al., 2023; Dey et al., 2023), was reflected in the development of prompt-based skills that utilize given prompts and LLMs to respond to the current context. Developers may create a prompt-

based generative distribution featuring their own prompts by copy-pasting several lines of code and configuration descriptions and selecting the generative services of interest as a parameter (more detailed instructions are provided in our tutorials and documentation). One of the main features of DeepPavlov Dream is a multi-skill support which allows a dialog system to contain and switch between several different prompts during a dialog session. More details in Section 7.

Knowledge Bases help the dialog system to generate meaningful responses that require world knowledge. Although generative models may learn some information from the training data, correct responses to factoid question require the dialog system to utilize an updating Knowledge Base. Knowledge Bases contain facts in graph or textual form which can be used as knowledge of the dialog assistant. DeepPavlov Dream utilizes graph and textual KBs for different annotators and skills, e.g., knowledge-based question answering (KBQA) (Evseev and Arkhipov, 2020), fact retrieval, paragraph-based open-domain question answering (ODQA). For AI assistants, one of the main required features is a custom ontology and a knowledge graph which allows to extract and store structured information about users. DeepPavlov Dream now integrates support of the custom KGs that allows to integrate corporate knowledge locally without any concerns on the data safety.

7 Prompt-based Generative AI Assistants

Since the emergence of LLMs, they have been used to tackle a variety of natural language tasks. The earlier and smaller models had to be fine-tuned for each specific task. However, very large models have shown a remarkable capacity of handling the same tasks few-shot and zero-shot using prompts, which are tokens appended or prepended to the model's as an instruction to "guide" its behavior.

When using LLMs in dialog modeling, it is crucially important to avoid insensitive or potentially harmful content and tailor the responses for the specific user needs. That is where prompt engineering allows to steer the model in the right direction with no need for fine-tuning.

DeepPavlov Dream provides an approach to building Generative AI Assistants with prompt-based control and lets the user develop their own ones with the use of prompt engineering. For that, the platform features Generative Skills, which en-

capsulate Generative Model Services used to locally run the LLMs of the user's choice or utilize external Generative API services. Each Generative Skill is controlled by one user-specified prompt, utilizes selected Generative Service and is built using Dialog Flow Framework to provide an opportunity for the developer to control the skill in a script-based manner. To build an assistant, which comes in a form of a custom Generative AI Assistant Distribution, several Generative Skills can be combined with help of Prompt Selector picking the most relevant prompts among presented and Response Selector managing the dialog. Prompt selection could be performed in different ways: simple ranking of prompt-context pairs, ranking of pairs of context and prompts goals extracted from prompts using LLMs, predicting with LLMs based on prompts descriptions.

In upcoming releases, we plan to enhance our skills by incorporating structured, vector-based, and RMT-based (Bulatov et al., 2022, 2023) representations of episodic and working memory based on the dialog state and conversation history stored within DeepPavlov Dream. Given the importance of supporting queries over documents and knowledge bases, future versions of DeepPavlov Dream will support LLM-driven interaction with external data sources like corporate databases. We also plan to add support for prompt-chaining to enable reasoning simulation to address more complex problems through planning and to facilitate development of the autonomous AI agents.

8 Multimodal Generative AI Assistants

Users' expectations from chatbots are rapidly increasing, so *multimodality*, which is operating with images, audio and video, is becoming an important direction in dialog systems' development. While the audio input can be converted to text almost without losing sense (except of intonations and emotions), received images may bring a key information to a dialog.

DeepPavlov Dream utilizes stateless paradigm, which means that components do not store any information about the dialog. The dialog state contains all the information and is forwarded through the full pipeline (partially, according to the given formatters) by the Agent component. Sending images or video between containers can be time-consuming, so we offer to use a special database storing images and videos and send file paths in

this database through the pipeline.

We implemented one of the most obvious ideas, which is to convert images to textual modality using image captioning models. Textual input replacement with an original image caption does not work as expected, so we created a scripted approach which extracts topics and entities from the caption, detects the type of the objects, and returns scripted responses for several particular types of objects, such as people, animals and food. We are also working on the skills responding to the user with images to provide a more engaging and human-like experience. In addition to that, we are working on enabling Socratic Models-like (Zeng et al., 2022) approach to address user tasks by running conversations between skills understanding different modalities.

9 Developer Experience

DeepPavlov Dream Platform enables developers to build their own multi-skill dialog systems. Despite of low-code/no-code trends for development tools, our platform requires its users to work with python and docker that will give them enough flexibility and customization opportunities.

Developer's path starts with repository on GitHub³. Developer needs a PC with Ubuntu or other Unix-like OS capable of running docker. To run heavy NLP components, locally dedicated GPUs are necessary. Any IDE can be utilized, however, VS Code is needed to use our DF Designer to create custom scenario-driven skills in a visual interface.

To create a custom distribution, developer can either make it by hands or use a single command from DeepPavlov dreamtools specifying a list of selected components. Developer can talk to the system by running a chat in a developer's mode in a command line, via Telegram or using web interface on "/chat" endpoint of the agent⁴.

One can utilize some components via proxy. Agent and database components are always run locally, while proxied components are light-weight containers. All components not present in the proxy YAML-file for the current distribution are run locally, which allows the developer to run and debug any new or existing component locally.

Here are the main opportunities for customizing a dialog system in DeepPavlov Dream: one

may combine a new dialog system with particular existing components, change the parameters of these components in configuration files, create new components from scratch or change the existing components.

To create a custom scenario-driven DFF skill and add it to the distribution, the developer can use a single command from DeepPavlov dreamtools. To visually design a custom scenario-driven DFF skill, the developer can use DF designer (refer to the Workshop video⁵ for detailed instructions).

Prompt-based Generative AI Assistants can be created by hands using a template distribution. One can add any number of prompt-based skills and customize them by prompt engineering and selecting Generative Services of interest. Multi-skill dialog management is handled automatically.

Debugging a complex distributed platform is always a challenge. In the DeepPavlov Dream Platform, one can use different techniques to successfully debug their Distribution. One can utilize DeepPavlov Agent's console to debug based on the entire dialog state, docker logging output to debug individual components, or POST requests to DeepPavlov Agent via Postman or similar tools.

The developer can build a custom user experience around their Dream-based multi-skill AI assistant or connect it to some of the existing channels, such as Amazon Alexa, to make it available for the end users. There is a Workshop Video⁶ available with detailed instructions.

We also provide the documentation site for DeepPavlov Dream⁷. The site provides access to comprehensive resources for building intelligent conversational assistants tailored to users' specific needs. The site offers extensive documentation, release notes, and detailed examples to facilitate the development of advanced conversational AI applications. The site also provides opportunities to join the community of developers leveraging DeepPavlov Dream to shape the future of conversational AI technology.

10 Conclusion

As complex conversational systems are becoming more and more popular, it is important to make the development process of such systems easier for researchers and developers. DeepPavlov Dream is

³<https://github.com/deeppavlov/dream>

⁴<http://0.0.0.0:4242/chat>

⁵<https://www.youtube.com/watch?v=WV1FV9VBh1g>

⁶https://www.youtube.com/watch?v=WAN_I10-M4M

⁷<https://dream.deeppavlov.ai/>

an open-source conversational platform designed to let the users develop their own complex dialog systems and access existing NLP components for feature extraction and classification of user utterances. The platform features a variety of skills developed with scenario-driven, retrieval, and generative approaches using modern NLP techniques including prompt-based generation. These skills are organized into ready-to-use distributions of the multi-skill AI assistants. DeepPavlov Dream allows to customize dialog systems at all levels. The platform was battle-tested during Amazon Alexa Prize 3 and 4 and is now providing core infrastructure to facilitate the development of multi-skill generative AI assistants for industry and academia.

We built a DeepPavlov Dream⁸ Platform's website⁹ including documentation, tutorials and useful links, chat with the demo distribution. We have published a series of articles about DeepPavlov Dream on Medium¹⁰. We also have a YouTube channel¹¹ where we publish video workshops and seminars. To communicate with our team, one can use our forum¹². Short demo video is available on YouTube¹³.

Acknowledgements

We express gratitude to all of current and past developers and researchers of DeepPavlov.ai for their contribution to the DeepPavlov Dream Platform. We are thankful to Denis Kuznetsov for providing DeepPavlov Dream with the state-machine-based skill engine powered by Dialog Flow Framework. We are also grateful to Anastasia Klowait and Mikhail Zamyatin for the DeepPavlov Dream web-site, and designer Irina Nikitenko for Figure 1. Special thanks to Yurii Kuratov and Idris Yusupov who made a major contribution to the development of the original DREAM Socialbot during the Alexa Prize Challenge 3 (Kuratov et al., 2020).

Limitations

The DeepPavlov Dream Platform contains English, Russian, and multilingual components. However, the multilingual generative model does not always respond in the same language.

DeepPavlov Dream covers open-domain hybrid conversational systems able to chat on any topic in addition to the commercial scenario-driven template-based chatbots. Therefore, we propose to use generative models. Generative-based distributions are able to respond to most of the dialog contexts with various replies while consume significant computational resources requiring at least a single GPU to run models like DialoGPT. Current distributions have a limited number of task-oriented skills, like Factoid QA and weather skill.

Ethics Statement

(1) This material is the authors' own original work, which at this stage of project development has not been previously published elsewhere. (2) The paper is not currently being considered for publication elsewhere. (3) The paper reflects the authors' own research and analysis in a truthful and complete manner. (4) We acknowledge that the use of the generative language models like DialoGPT and others in production might lead to potential harm to the end user experience; while we have adopted measures to prevent inappropriate language output we can not guarantee that the dialog systems that incorporated generative models can be free of inappropriate language. (5) All conversations users have with the publicly deployed English distribution of DeepPavlov Dream available at¹⁴ are recorded and are available for the conversational AI researchers via¹⁵ as an open-source dialog dataset. Users have to agree to a privacy agreement prior to talking to the Dream distribution. They are warned that their dialogs will become publicly available as part of the dataset and are strongly encouraged to never share personal details with the Dream distribution. (6) The dialogs that developers have with the Dream distribution running completely via proxy (including MongoDB instance used by DeepPavlov Agent) will also be available publicly, while the dialogs between the developer and distributions that run at least DeepPavlov Agent with MongoDB instance locally will not become available as a part of our open-source conversational AI dataset.

References

Linto AI. 2022a. Linto ai. <https://linto.ai/>.

¹⁴<https://dream.deeppavlov.ai/>

¹⁵<http://deeppavlov.ai/dream/datasets/>

⁸<https://github.com/deeppavlov/dream>

⁹<https://dream.deeppavlov.ai/>

¹⁰<https://medium.com/deeppavlov>

¹¹<https://www.youtube.com/c/DeepPavlov>

¹²<https://forum.deeppavlov.ai/>

¹³<https://youtu.be/1EwiqNsfoMI>

- Mycroft AI. 2022b. Mycroft ai. <https://mycroft.ai/>.
- Amazon. 2022. Amazon lex. <https://docs.aws.amazon.com/lex/index.html>.
- Amelia. 2022. Amelia. <https://amelia.ai/conversational-ai/>.
- Avaamo. 2022. Avaamo. <https://avaamo.ai/conversational-ai-platform/>.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- D Baymurzina, Yu Kuratov, D Kuznetsov, D Kornev, and M Burtsev. 2021a. Evaluation of conversational skills for commonsense.
- Dilyara Baymurzina, Denis Kuznetsov, Dmitry Evseev, Dmitry Karpov, Alsu Sagirova, Anton Peganov, Fedor Ignatov, Elena Ermakova, Daniil Cherniavskii, Sergey Kumeiko, et al. 2021b. Dream technical report for the alexa prize 4. *4th Proceedings of Alexa Prize*.
- Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.
- Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*.
- Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.
- Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nikolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yurii Kuratov, Denis Kuznetsov, et al. 2018. Deepavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127.
- Anastasia Chizhikova, Vasily Konovalov, and Mikhail Burtsev. 2023. Multilingual case-insensitive named entity recognition. In *Advances in Neural Computation, Machine Learning, and Cognitive Research VI*, pages 448–454, Cham. Springer International Publishing.
- DeepPavlov. 2023. Deepavlov dream. <https://dream.deepavlov.ai/>.
- Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. 2023. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*.
- DA Evseev. Lightweight and accurate system for entity extraction and linking.
- DA Evseev and M Yu Arkhipov. 2020. Sparql query generation for complex question answering with bert and bilstm-based model. In *Computational Linguistics and Intellectual Technologies*, pages 270–282.
- DA Evseev, MS Nagovitsin, and DP Kuznetsov. Controllable multi-attribute dialog generation with pals and grounding knowledge.
- Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and Bill Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. *arXiv preprint arXiv:2009.06978*.
- Google. 2022. Google dialogflow. <https://cloud.google.com/dialogflow>.
- Denis Kapelyushnik, Dilyara Baymurzina, Denis Kuznetsov, and Mikhail Burtsev. 2022. Automatic generation of conversational skills from dialog datasets. In *Advances in Neural Computation, Machine Learning, and Cognitive Research VI: Selected Papers from the XXIV International Conference on Neuroinformatics, October 17-21, 2022, Moscow, Russia*, pages 31–41. Springer.
- Dmitry Karpov and Mikhail Burtsev. 2023. Monolingual and cross-lingual knowledge transfer for topic classification. *Proceedings of AINL 2023*.
- Dmitry Karpov and Vasily Konovalov. 2023. Knowledge transfer in the multi-task encoder-agnostic transformer-based models (). *Komp'yuternaja Lingvistika i Intellektual'nye Tehnologii*.
- Alina Kolesnikova, Yuri Kuratov, Vasily Konovalov, and Mikhail Burtsev. 2022. Knowledge distillation of russian language models with reduction of vocabulary. *arXiv preprint arXiv:2205.02340*.
- Vasily Konovalov, Pavel Gulyaev, Alexey Sorokin, Yury Kuratov, and Mikhail Burtsev. 2020. Exploring the bert cross-lingual transfer for reading comprehension. In *Dialogue-21*.
- Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations—democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*.
- Kore.ai. 2022. Kore.ai. <https://kore.ai/>.
- Yuri Kuratov, Idris Yusupov, Dilyara Baymurzina, Denis Kuznetsov, Daniil Cherniavskii, Alexander Dmitrievskiy, Elena Ermakova, Fedor Ignatov,

- Dmitry Karpov, Daniel Kornev, et al. 2020. Dream technical report for the alexa prize 2019. *Alexa Prize Proceedings*.
- Denis Kuznetsov, Dmitry Evseev, Lidia Ostyakova, Oleg Serikov, Daniel Kornev, and Mikhail Burtsev. 2021. Discourse-driven integrated dialogue development environment for open-domain dialogue systems. In *Proceedings of the 2nd Workshop on Computational Approaches to Discourse*, pages 29–51.
- TA Le, MA Petrov, YM Kuratov, and MS Burtsev. 2019. Sentence level representation and language models in the task of coreference resolution for russian. *Computational Linguistics and Intellectual Technologies*, pages 364–373.
- Lidiia Ostyakova, M Molchanova, Ksenia Petukhova, Nika Smilga, D Kornev, and M Burtsev. 2022. Corpus with speech function annotation: Challenges, advantages, and limitations. *Computational Linguistics and Intellectual Technologies*, pages 1129–1139.
- RASA. 2022. Rasa platform. <https://rasa.com/product/rasa-platform/>.
- Beksultan Sagyndyk, Dilyara Baymurzina, and Mikhail Burtsev. 2022. Deeppavlov topics: Topic classification dataset for conversational domain in english. In *Studies in Computational Intelligence, Springer*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.
- Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.
- IBM Watson. 2022. Ibm watson assistant. <https://www.ibm.com/products/watson-assistant>.
- Dian Yu and Zhou Yu. 2019. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. *arXiv preprint arXiv:1908.10023*.
- Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. 2022. [Socratic models: Composing zero-shot multimodal reasoning with language](#).

Author Index

- Agarwal, Anshul, 103
Agarwal, Shantanu, 247
Akiki, Christopher, 304, 588
Althammer, Sophia, 581
Alva-Manchego, Fernando, 86
Amini, Afra, 463
Antoniak, Maria, 377
Artzi, Yoav, 412
- Bach, Stephen, 479
Baranchuk, Dmitry, 558
Barry, Joel, 247
Basile, Valerio, 348
Bast, Hannah, 389
Baumgärtner, Tim, 569
Behera, Laxmidhar, 103
Belkada, Younes, 558
Berrebbi, Dan, 400
Bhat, Riyaz, 51
Bi, Wei, 508
Biderman, Stella, 588
Biemann, Chris, 328
Bioglio, Livio, 348
Bornea, Mihaela, 51
Borzunov, Alexander, 558
Bosca, Alessio, 348
Boschee, Elizabeth, 247
Bosco, Cristina, 348
Braslavski, Pavel, 524
Bross, Juergen, 51
Brown, Susan Windisch, 1
Buchmann, Jan, 291
Burtsev, Mikhail, 599
- Cai, Deng, 508
Callison-Burch, Chris, 1
Camacho-Collados, Jose, 86
Cambria, Erik, 127
Cao, Shulin, 179
Chan, Aaron, 264
Chan, Robin, 463
Chau, Duen Horng, 516
Che, Wanxiang, 95
Chen, Chen, 217
Chen, Jindong, 456
Chen, Qiguang, 95
Chen, Sihong, 217
Chen, Xiaoshuai, 217
- Chen, Xiayu, 456
Chen, Yiren, 217
CHEN, YUANYONG, 179
Chen, Yuanzhe, 471
Chumachenko, Artem, 558
Cohan, Arman, 336
Crego, Josep, 369
Cui, Leyang, 508
Cui, Wenjuan, 76
Cui, Xin, 499
- Dabre, Raj, 257
Dalmia, Siddharth, 400
Dalvi, Fahim, 226
de Gibert, Ona, 315
Dettmers, Tim, 558
Dibia, Victor, 113
Ding, Ning, 274
Dror, Rotem, 1
Du, Xiaoyong, 217
Du, Yi, 76
Duh, Kevin, 161
Duong, Khoa, 549
Dupont, Gérard, 304
Durrani, Nadir, 226
Dusek, Ondrej, 444
Dycke, Nils, 291
Dymetman, Marc, 144
- Eiser, Isabel, 328
El-Assady, Mennatallah, 463
Eo, Sugyeong, 190
Evseev, Dmitry, 599
- Fadnis, Kshitij, 51
Fang, Haishuo, 569
Feldhus, Nils, 421
Feng, Jiangtao, 489
Feng, Yunlong, 95
Fernandes, Patrick, 400
Field, Anjalie, 377
Fincke, Steven, 247
Fischer, Tim, 328
Florian, Radu, 51
Franz, Martin, 51
Fu, Xiaojin, 471
Fuchs, Leopold, 357

Garanina, Ekaterina, 444
 Ge, Mengshi, 127
 Goldberg, Yoav, 282
 Gou, Weigang, 217
 Goyal, Pawan, 103
 Gu, Nianlong, 235
 Gu, Shiyu, 11
 Guan, Jian, 499
 Gul, Mustafa Omer, 412
 Guo, Han, 217
 Guo, Minghao, 336
 Gurevych, Iryna, 291, 569

 Hahnloser, Richard H.R., 235
 Han, Wenjuan, 11
 Han, Xiongwei, 471
 Hayashi, Tomoki, 400
 He, Bingru, 11
 He, Kai, 127
 Hertel, Matthias, 389
 Hira, Moto, 400
 Hoang, Thanh Lam, 357
 Hofstätter, Sebastian, 581
 Hohman, Fred, 516
 Hoi, Steven C.H., 31
 Hou, Cheng, 217
 Hou, Lei, 179
 Hou, Zhaoyi, 1
 Hu, Shengding, 274
 Huang, Chenyan, 508
 Huang, Guoping, 508
 Huang, Mengyi, 76
 Huang, Minlie, 499
 Huang, Shan, 217
 Huang, Xinting, 508

 Ignatov, Fedor, 599
 Ikeda, Masami, 549
 Inaguma, Hirofumi, 400
 Itoh, Mari, 549
 Iyer, Bhavani, 51

 Jang, Yoonna, 190
 Jenkins, Chris, 247
 Jernite, Yacine, 304
 Ji, Heng, 1
 Ji, Jinhao, 11
 Jiang, Haiyun, 508
 Jin, Hailong, 179
 Joshi, Brihi, 264

 Kadakia, Akshen, 264
 Kang, Zhanhui, 217
 Kanojia, Diptesh, 257
 Karpov, Dmitry, 599
 Kasner, Zdeněk, 444
 Kawahara, Daisuke, 538
 Kiyomaru, Hirokazu, 538
 Klein, Lauren, 377
 Koch, Gertraud, 328
 Kodama, Takashi, 538
 Kornev, Daniel, 599
 Kosenko, Dmitry, 599
 Kruszewski, Germán, 144
 Kumar, Vishwajeet, 51
 Kuroda, Masakata, 549
 Kurohashi, Sadao, 538
 Kuznetsov, Iliia, 291
 Kwon, Bum Chul, 42

 Landwehr, Fabian, 208
 Laurençon, Hugo, 304
 Le, Hung, 31
 Lee, Dong-Ho, 264
 Lee, Jungseob, 190
 Lee, Seounghoon, 190
 Lee, Seugnjun, 190
 Li, Dongxu, 31
 Li, Feifei, 217
 Li, Juanzi, 179
 Li, Junnan, 31
 Li, Piji, 508
 Li, Ren, 471
 Li, Sha, 1
 Li, Shiqian, 11
 Li, Xiao, 127
 Li, Xiaobing, 63
 Li, Xiaorui, 471
 Li, Xinyang, 11
 Li, Yudong, 217
 Li, Yulong, 51
 Li, Zhen, 11
 Lim, Heuseok, 190
 Lin, Jimmy, 588
 Liu, Bo, 499
 Liu, Haoyan, 217
 Liu, Jiafeng, 63
 Liu, Jiawen, 11
 Liu, Lijuan, 456
 Liu, Liquan, 217
 Liu, Sijia, 11
 Liu, Weijie, 217

Liu, Zhiyuan, 274
 Liu, Ziyi, 264
 Lopez, Vanessa, 357
 Luccioni, Sasha, 304
 Lv, Xin, 179
 Lv, Xingtai, 274
 Lyu, Yidong, 11

Ma, Yuxi, 11
 Maiti, Soumi, 400
 Malykh, Valentin, 524
 Mao, Kun, 471
 Mao, Rui, 127
 Mao, Weiquan, 217
 Martin, Lara, 1
 Martinez Galindo, Marcos, 357
 Mascarell, Laura, 208
 McCarley, Scott, 51
 McNamee, Paul, 161
 Mi, Boyu, 336
 Mihindikulasooriya, Nandana, 42
 Mitani, Ryosuke, 264
 Moon, Hyeonseok, 190
 Morstatter, Fred, 436
 Mostajabdaveh, Mahdi, 471
 Mun, Jimin, 377
 Murawaki, Yugo, 538

Nagano, Nozomi, 549
 Nan, Linyong, 336
 Narahari, Kiran, 264
 Natsume-Kitatani, Yayoi, 549
 Ni, Zhaoheng, 400
 Niu, Xinyi, 11

Ogundepo, Odunayo, 588
 Oladipo, Akintunde, 588
 Omura, Kazumasa, 538

Palmer, Martha, 1
 Panchenko, Alexander, 524
 Park, Chanjun, 190
 Patti, Viviana, 348
 Peng, Yifan, 400
 Petersen-Frey, Fynn, 328
 Petukhova, Ksenya, 599
 Picco, Gabriele, 357
 Piktus, Aleksandra, 304, 588
 Pino, Juan, 400
 Platek, Ondrej, 444
 Polak, Peter, 400

Potthast, Martin, 588
 Priniski, John, 436
 Puerto, Haritz, 569
 Pujara, Jay, 264
 Purpura, Alberto, 357

Qi, Zhenting, 336
 Qiao, Yu, 489
 Qin, Libo, 95
 Qiu, Lin, 11

Radev, Dragomir, 336
 Raffel, Colin, 558
 Ramamonjison, Rindra, 471
 Ran, Yu, 499
 Razzhigaev, Anton, 524
 Ren, Jiaxuan, 1
 Ren, Xiang, 264
 Riabinin, Maksim, 558
 Rogers, Anna, 304
 Rosenthal, Sara, 51
 Roukos, Salim, 51
 Rozen, Jos, 144
 Ruan, Lecheng, 11
 Ruan, Ning, 456

Sachdeva, Rachneet, 569
 Sajjad, Hassan, 226
 Salnikov, Mikhail, 524
 Samygin, Pavel, 558
 Sandhan, Jivnesh, 103
 Sandhan, Tushar, 103
 Sap, Maarten, 377
 Sarti, Gabriele, 421
 savarese, silvio, 31
 Sawant, Chinmay, 257
 Schneider, Florian, 328
 Schoelkopf, Hailey, 588
 Sekiya, Toshiyuki, 264
 Sen, Jaydeep, 51
 Seo, Jaehyung, 190
 Sertkan, Mete, 581
 Sharf, Jacob, 412
 Shen, Linlin, 217
 Shi, Jiatong, 400
 Shi, Shuming, 508
 Shi, Wenhong, 217
 Shi, Yu-Zhe, 11
 Shibuya, Takashi, 264
 Shishkina, Yana, 599
 Shu, Lei, 456

Sickert, Ludwig, 421
 Sil, Avi, 51
 Smilga, Veronika, 599
 Sohrab, Mohammad Golam, 549
 Song, Dongze, 76
 Song, Kaiqiang, 508
 Song, Xinying, 456
 Steinmann, Thomas, 208
 Suchocki, Reece, 1
 Sultan, Md Arafat, 51
 Sumita, Eiichiro, 169, 257
 Sun, Maosong, 63, 274
 Sun, Ningyuan, 217
 Sun, Xingwu, 217

 Takamura, Hiroya, 549
 Talimanchuk, Maxim, 599
 Tan, Bowen, 456
 Tang, Duyu, 508
 Tang, Yun, 400
 Tariverdian, Sewin, 569
 Taub-Tabib, Hillel, 282
 Tham, Isaac, 1
 Tian, Rong, 217
 Tiedemann, Jörg, 315
 Topić, Goran, 549

 Ueda, Nobuhiro, 538
 Ushio, Asahi, 86
 Utiyama, Masao, 169

 van der Wal, Oskar, 421
 Vanderlyn, Lindsey, 136
 Verma, Ishaan, 436
 Villegas, Paulo, 304
 Vu, Ngoc Thang, 136
 Väth, Dirk, 136

 Walsh, Melanie, 377
 Walter, Sebastian, 389
 Wang, Guangsen, 31
 Wang, Kexin, 569
 Wang, Longyue, 508
 Wang, Ludi, 76
 Wang, Maria, 456
 Wang, Xiaolin, 169
 Wang, Yan, 508
 Wang, Yaoxiang, 489
 Wang, Zijie J., 516
 Watanabe, Shinji, 400
 Wu, Taiqiang, 217
 Wu, Zhenyu, 489

 Wu, Zhiyong, 489

 Xin, Amy, 179
 Xing, Linzi, 471
 Xu, Hainiu, 1
 xu, jianjun, 179
 Xu, Jingjing, 489
 Xu, Jitao, 369
 Xu, Qiao, 11
 Xu, Xiao, 95
 Xu, Yifan, 11

 Yahya, Bernardo, 190
 Yair, Itay, 282
 Yan, Brian, 400
 Yan, Kimmo, 217
 Yao, Zijun, 179
 Ye, Jiacheng, 489
 Yu, Jifan, 179
 Yu, Peilin, 479
 Yu, Timothy, 471
 Yvon, François, 369

 Zhang, Hao, 569
 Zhang, Li, 1
 Zhang, Peng, 179
 Zhang, Rong, 51
 Zhang, Tianyi, 1
 Zhang, Xiaohui, 400
 Zhang, Xinran, 63
 Zhang, Xinyu, 588
 Zhang, Xuan, 161
 Zhang, Yong, 471
 Zhang, Zhen, 274
 Zhang, Zhexin, 499
 Zhao, Enbo, 508
 Zhao, Jing, 217
 Zhao, Weilin, 274
 Zhao, Xinyu, 11
 Zhao, Yilun, 336
 Zhao, Zhe, 217
 Zhao, Zijian, 11
 Zharikova, Diliara, 599
 Zhou, Leon, 1
 Zhou, Yuanchun, 76
 Zhu, Tao, 217
 Zhu, Yixin, 11
 Zhu, Yun, 456
 Zyska, Dennis, 291