

EMNLP 2023

**The 2023 Conference on Empirical Methods in Natural  
Language Processing**

**Proceedings of the System Demonstrations**

December 6-10, 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](mailto:acl@aclweb.org)

ISBN 978-8-89176-067-7

## **Introduction**

Welcome to the proceedings of the system demonstration track of the 2023 Conference on Empirical Methods in Natural Language Processing on December 6th – December 10th, 2023. For the EMNLP 2023 system demonstration track, we received a record number of 203 submissions, of which 52 were selected for inclusion in the program (acceptance rate of 25%) 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 EMNLP conference is a hybrid event. The demonstration papers will be presented through pre-recorded talks and in presence during the poster sessions.

Yansong Feng and Els Lefever  
EMNLP 2023 System Demonstration Chairs

# Organizing Committee

## Demonstration Chairs

Yansong Feng, Peking University, China

Els Lefever, LT3, Ghent University, Belgium

# Program Committee

## Best Demo Paper Award Committee

Walter Daelemans, University of Antwerp  
Yusuke Miyao, the University of Tokyo  
Djamé Seddah, Inria Paris

## Program Committee

Alan Akbik, Humboldt-Universität zu Berlin  
Ekaterina Artemova, Toloka.AI  
Giosuè Baggio, Norwegian University of Science and Technology  
Ayoub Bagheri, Department of Methodology and Statistics, Utrecht University  
Timothy Baldwin, MBZUAI  
Denilson Barbosa, University of Alberta  
Farah Benamara, University of toulouse  
Gábor Berend, University Of Szeged  
Marianna Bolognesi, University of Bologna  
Francis Bond, Palacký University  
Georgeta Bordea, Université de Bordeaux  
Johan Bos, University of Groningen  
Elena Cabrio, Université Côte d'Azur, Inria, CNRS, I3S  
Yi Cai, South China University of Technology  
Deng Cai, The Chinese University of Hong Kong  
Jose Camacho-Collados, Cardiff University  
Hejing Cao, Peking University  
Tommaso Caselli, Rijksuniversiteit Groningen  
Yekun Chai, Baidu  
Franklin Chang, Kobe City University of Foreign Studies  
Wanxiang Che, Harbin Institute of Technology  
Chung-Chi Chen, National Institute of Advanced Industrial Science and Technology  
Guanyi Chen, Central China Normal University  
Kehai Chen, School of Computer Science and Technology, Harbin Institute of Technology  
Sanyuan Chen, Harbin Institute of Technology  
Xinchi Chen, Amazon AWS  
Yubo Chen, Institute of Automation, Chinese Academy of Sciences  
Prafulla Kumar Choubey, Salesforce AI Research  
Bonaventura Coppola, University of Trento  
Heriberto Cuayahuitl, University of Lincoln  
Luna De Bruyne, CLiPS, University of Antwerp  
Orphee De Clercq, LT3, Ghent University  
Xiang Deng, The Ohio State University  
Luigi Di Caro, University of Turin  
Jakub Dotlacil, Utrecht University  
Zhicheng Dou, Renmin University of China  
Zi-Yi Dou, UCLA  
Aleksandr Drozd, RIKEN Center for Computational Science  
Haowei Du, Peking University  
Steffen Eger, NLLG Lab, Bielefeld University

Stefano Faralli, University of Rome Sapienza  
Jiazhan Feng, Peking University  
Siva Reddy Gangireddy, SoapBox Labs  
Voula Giouli, Institute for Language & Speech Processing, ATHENA Research & Innovation Centre  
Udo Hahn, Friedrich-Schiller-Universitaet Jena  
Samar Haider, University of Pennsylvania  
Lifeng Han, The University of Manchester  
Wenjuan Han, Beijing Jiaotong University  
Sanda Harabagiu, University of Texas at Dallas  
Iris Hendrickx, Centre for Language Studies, Radboud University Nijmegen  
Delia Irazu Hernandez Farias, Instituto Nacional de Astrofisica, Optica y Electronica  
Xudong Hong, Saarland University / MPI Informatics  
Veronique Hoste, LT3, Ghent University  
Baotian Hu, Harbin Institue of Technology, Shenzhen  
Yutong Hu, Peking University  
Xinyu Hua, Bloomberg  
Quzhe Huang, Peking University  
Shujian Huang, National Key Laboratory for Novel Software Technology, Nanjing University  
Nancy Ide, Vassar College/Brandeis University  
Joseph Marvin Imperial, University of Bath  
Radu Tudor Ionescu, University of Bucharest  
Yichen Jiang, University of North Carolina at Chapel Hill  
Elena Karajosova, Freie Universität  
Mladen Karan, Queen Mary University  
Yoshihide Kato, Nagoya University  
Yova Kementchedjhieva, University of Copenhagen  
Salam Khalifa, Stony Brook University  
Halil Kilicoglu, University of Illinois at Urbana-Champaign  
Dong-Jin Kim, Hanyang University  
Mamoru Komachi, Hitotsubashi University  
Rik Koncel-Kedziorski, Kensho Technologies  
Maarit Koponen, University of Eastern Finland  
Viet Lai, University of Oregon  
Yuxuan Lai, the Open University of China  
Kysong Lee, SOCO AI  
Manling Li, Northwestern University  
Maoxi Li, School of Computer Information Engineering, Jiangxi Normal University  
Xintong Li, Apple  
Zuchao Li, Wuhan University  
Jiangming Liu, Yunnan University  
Tianyu Liu, Peking University  
Xiao Liu, Microsoft Research Asia  
Yang Janet Liu, Georgetown University  
Aaron Maladry, Ghent University  
Alda Mari, <http://www.institutnicod.org/>  
Ilia Markov, Vrije Universiteit Amsterdam, CLTL  
Eugenio Martínez Cámara, University of Jaén  
John P. McCrae, Insight Center for Data Analytics, National University of Ireland Galway  
Pushkar Mishra, Meta AI  
Amita Misra, Amazon

Richard Moot, CNRS  
Gaku Morio, Research & Development Group, Hitachi America, Ltd.  
Huy Nguyen, Amazon  
Yixin Nie, Meta  
Robert Östling, Department of Linguistics, Stockholm University  
Jessica Ouyang, University of Texas at Dallas  
Alexander Panchenko, Skolkovo Institute of Science and Technology  
Jiaxin Pei, University of Michigan  
Massimo Poesio, Queen Mary University of London  
Octavian Popescu, IBM Yorktown Research  
Adithya Pratapa, Carnegie Mellon University  
Anil Ramakrishna, Amazon  
Julia Rayz, Purdue University  
Steffen Remus, Hamburg University  
Alexander Rush, Cornell University  
Magnus Sahlgren, AI Sweden  
Shuming Shi, Tencent AI Lab  
Lei Shu, Google Research  
Jyotika Singh, Placemakr  
Pranaydeep Singh, LT3, University of Ghent  
Ionut-Teodor Sorodoc, Amazon  
Mark Steedman, University of Edinburgh  
Carlo Strapparava, FBK-irst  
Lixin Su, Institute of Computing Technology, Chinese Academy of Sciences  
Chengjie Sun, Harbin Institute of Technology  
Kai Sun, Meta  
Marek Šuppa, Comenius University in Bratislava  
Colin Swaelens, Ghent University  
Aarne Talman, University of Helsinki  
Chongyang Tao, Microsoft Corporation  
Mingxu Tao, Peking University  
Irina Temnikova, Big Data for Smart Society Institute (GATE), Bulgaria  
Arda Tezcan, LT3, Language and Translation Technology Team, Ghent University  
Christoph Tillmann, IBM Research  
Marco Turchi, Zoom Video Communications  
Rik van Noord, University of Groningen  
Bram Vanroy, KU Leuven  
Di Wang, ContextLogic Inc  
Ke Wang, Huawei Technologies Ltd.  
Qingyun Wang, University of Illinois at Urbana-Champaign  
Xun Wang, Microsoft  
Wei Wang, Apple AI/ML  
Martijn Wieling, University of Groningen  
Yang Wu, Harbin Institute of Technology  
Zirui Wu, Peking University  
Jiarong Xu, Fudan University  
Jingjing Xu, Shanghai AI Lab  
Ruo Chen Xu, Microsoft  
Silei Xu, Stanford University  
Wei Xu, Georgia Institute of Technology  
Yan Xu, Hong Kong University of Science and Technology

Jingfeng Yang, Amazon  
Yaqin Yang, Paypal  
Shuang (Sophie) Zhai, University of Oklahoma  
Biao Zhang, Google Research  
Chen Zhang, Peking University  
Meishan Zhang, Harbin Institute of Technology (Shenzhen), China  
Tianlin Zhang, The University of Manchester  
Wen Zhang, Zhejiang University  
Jie Zhao, Microsoft  
Xiang Zhao, National University of Defense Technology  
Ming Zhong, University of Illinois Urbana-Champaign  
Tiantian Zhu, Harbin Institute of Technology (Shenzhen)



## Table of Contents

|   |     |
|---|-----|
| <i>Fabricator: An Open Source Toolkit for Generating Labeled Training Data with Teacher LLMs</i><br>Jonas Golde, Patrick Haller, Felix Hamborg, Julian Risch and Alan Akbik .....   | 1   |
| <i>End-to-End Evaluation for Low-Latency Simultaneous Speech Translation</i><br>Christian Huber, Tu Anh Dinh, Carlos Mullov, Ngoc-Quan Pham, Thai Binh Nguyen, Fabian Retkowsky, Stefan Constantin, Enes Ugan, Danni Liu, Zhaolin Li, Sai Koneru, Jan Niehues and Alexander Waibel .....                                    | 12  |
| <i>CHATREPORT: Democratizing Sustainability Disclosure Analysis through LLM-based Tools</i><br>Jingwei Ni, Julia Bingler, Chiara Colesanti-Senni, Mathias Kraus, Glen Gostlow, Tobias Schimanski, Dominik Stambach, Saeid Ashraf Vaghefi, Qian Wang, Nicolas Webersinke, Tobias Wekhof, Tingyu Yu and Markus Leippold ..... | 21  |
| <i>RaLLe: A Framework for Developing and Evaluating Retrieval-Augmented Large Language Models</i><br>Yasuto Hoshi, Daisuke Miyashita, Youyang Ng, Kento Tatsuno, Yasuhiro Morioka, Osamu Torii and Jun Deguchi .....  | 52  |
| <i>VIST5: An Adaptive, Retrieval-Augmented Language Model for Visualization-oriented Dialog</i><br>Henrik Voigt, Nuno Carvalhais, Monique Meuschke, Markus Reichstein, Sina Zarrie and Kai Lawonn .....   | 70  |
| <i>H2O Open Ecosystem for State-of-the-art Large Language Models</i><br>Arno Candell, Jon McKinney, Philipp Singer, Pascal Pfeiffer, Maximilian Jeblick, Chun Ming Lee and Marcos Conde .....   | 82  |
| <i>Koala: An Index for Quantifying Overlaps with Pre-training Corpora</i><br>Thuy-Trang Vu, Xuanli He, Gholamreza Haffari and Ehsan Shareghi .....  | 90  |
| <i>Sudowoodo: A Chinese Lyric Imitation System with Source Lyrics</i><br>Yongzhu Chang, Rongsheng Zhang, Lin Jiang, Qihang Chen, Le Zhang and Jiashu Pu .....   | 99  |
| <i>ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format</i><br>Qi Zhu, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, Baolin Peng, Zheng Zhang, Shutong Feng, Michael Heck, Nurul Lubis, Dazhen Wan, Xiaochen Zhu, Jianfeng Gao, Milica Gasic and Minlie Huang .....                      | 106 |
| <i>FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge</i><br>Farima Fatahi Bayat, Kun Qian, Benjamin Han, Yisi Sang, Anton Belyy, Samira Khorshidi, Fei Wu, Ihab Ilyas and Yunyao Li .....   | 124 |
| <i>YATO: Yet Another deep learning based Text analysis Open toolkit</i><br>Zeqiang Wang, Yile Wang, Jiageng Wu, Zhiyang Teng and Jie Yang .....   | 131 |
| <i>Spacerini: Plug-and-play Search Engines with Pyserini and Hugging Face</i><br>Christopher Akiki, Odunayo Ogundepo, Aleksandra Piktus, Xinyu Zhang, Akintunde Oladipo, Jimmy Lin and Martin Potthast .....  | 140 |
| <i>Adapters: A Unified Library for Parameter-Efficient and Modular Transfer Learning</i><br>Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vuli, Sebastian Ruder, Iryna Gurevych and Jonas Pfeiffer .....   | 149 |
| <i>INTELMO: Enhancing Models' Adoption of Interactive Interfaces</i><br>Chunxu Yang, Chien-Sheng Wu, Lidiya Murakhovska, Philippe Laban and Xiang Anthony Chen  | 161 |

|   |     |
|---|-----|
| <i>Humanoid Agents: Platform for Simulating Human-like Generative Agents</i><br>Zhilin Wang, Yu Ying Chiu and Yu Cheung Chiu .....  | 167 |
| <i>TP-Detector: Detecting Turning Points in the Engineering Process of Large-scale Projects</i><br>Qi Wu, WenHan Chao, Xian Zhou and Zhunchen Luo .....   | 177 |
| <i>CLEVA: Chinese Language Models EVALuation Platform</i><br>Yanyang Li, Jianqiao Zhao, Duo Zheng, Zi-Yuan Hu, Zhi Chen, Xiaohui Su, Yongfeng Huang,<br>Shijia Huang, Dahua Lin, Michael Lyu and Liwei Wang .....   | 186 |
| <i>DOPA METER – A Tool Suite for Metrical Document Profiling and Aggregation</i><br>Christina Lohr and Udo Hahn .....   | 218 |
| <i>Muted: Multilingual Targeted Offensive Speech Identification and Visualization</i><br>Christoph Tillmann, Aashka Trivedi, Sara Rosenthal, Santosh Borse, Rong Zhang, Avirup Sil and<br>Bishwaranjan Bhattacharjee .....  | 229 |
| <i>Gentopia.AI: A Collaborative Platform for Tool-Augmented LLMs</i><br>Binfeng Xu, Xukun Liu, Hua Shen, Zeyu Han, Yuhan Li, Murong Yue, Zhiyuan Peng, Yuchen<br>Liu, Ziyu Yao and Dongkuan Xu .....  | 237 |
| <i>MusicAgent: An AI Agent for Music Understanding and Generation with Large Language Models</i><br>Dingyao Yu, Kaitao Song, Peiling Lu, Tianyu He, Xu Tan, Wei Ye, Shikun Zhang and Jiang Bian<br>246  |     |
| <i>SentAlign: Accurate and Scalable Sentence Alignment</i><br>Steinthor Steingrímsson, Hrafn Loftsson and Andy Way .....  | 256 |
| <i>QACheck: A Demonstration System for Question-Guided Multi-Hop Fact-Checking</i><br>Liangming Pan, Xinyuan Lu, Min-Yen Kan and Preslav Nakov .....  | 264 |
| <i>RobustQA: A Framework for Adversarial Text Generation Analysis on Question Answering Systems</i><br>Yasaman Boreshban, Seyed Morteza Mirbostani, Seyedeh Fatemeh Ahmadi, Gita Shojaei, Fate-<br>meh Kamani, Gholamreza Ghassem-Sani and Seyed Abolghasem Mirroshandel .....          | 274 |
| <i>Kandinsky: An Improved Text-to-Image Synthesis with Image Prior and Latent Diffusion</i><br>Anton Razzhigaev, Arseniy Shakhmatov, Anastasia Maltseva, Vladimir Arkhipkin, Igor Pavlov,<br>Ilya Ryabov, Angelina Kuts, Alexander Panchenko, Andrey Kuznetsov and Denis Dimitrov ..... | 286 |
| <i>NewsRecLib: A PyTorch-Lightning Library for Neural News Recommendation</i><br>Andreea Iana, Goran Glavaš and Heiko Paulheim .....  | 296 |
| <i>MiniChain: A Small Library for Coding with Large Language Models</i><br>Alexander Rush .....   | 311 |
| <i>Okapi: Instruction-tuned Large Language Models in Multiple Languages with Reinforcement Learning<br/>from Human Feedback</i><br>Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi and Thien<br>Nguyen .....  | 318 |
| <i>SAGEViz: Schema GEneration and Visualization</i><br>Sugam Devare, Mahnaz Koupaee, Gautham Gunapati, Sayontan Ghosh, Sai Vallurupalli, Yash<br>Kumar Lal, Francis Ferraro, Nathanael Chambers, Greg Durrett, Raymond Mooney, Katrin Erk and<br>Niranjan Balasubramanian .....         | 328 |
| <i>Thresh: A Unified, Customizable and Deployable Platform for Fine-Grained Text Evaluation</i><br>David Heineman, Yao Dou and Wei Xu .....   | 336 |

|   |     |
|---|-----|
| <i>InsightPilot: An LLM-Empowered Automated Data Exploration System</i><br>Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han and Dongmei Zhang .....  | 346 |
| <i>SynJax: Structured Probability Distributions for JAX</i><br>Miloš Stanojević and Laurent Sartran .....   | 353 |
| <i>RESIN-EDITOR: A Schema-guided Hierarchical Event Graph Visualizer and Editor</i><br>Khanh Duy Nguyen, Zixuan Zhang, Reece Suchocki, Sha Li, Martha Palmer, Susan Windisch Brown, Jiawei Han and Heng Ji .....  | 365 |
| <i>DRGCoder: Explainable Clinical Coding for the Early Prediction of Diagnostic-Related Groups</i><br>Daniel Hajjaligol, Derek Kaknes, Tanner Barbour, Daphne Yao, Chris North, Jimeng Sun, David Liem and Xuan Wang .....  | 373 |
| <i>CAMRA: Copilot for AMR Annotation</i><br>Jon Cai, Shafiuddin Rehan Ahmed, Julia Bonn, Kristin Wright-Bettner, Martha Palmer and James H. Martin .....  | 381 |
| <i>Reaction Miner: An Integrated System for Chemical Reaction Extraction from Textual Data</i><br>Ming Zhong, Siru Ouyang, Yizhu Jiao, Priyanka Kargupta, Leo Luo, Yanzhen Shen, Bobby Zhou, Xianrui Zhong, Xuan Liu, Hongxiang Li, Jinfeng Xiao, Minhao Jiang, Vivian Hu, Xuan Wang, Heng Ji, Martin Burke, Huimin Zhao and Jiawei Han .....   | 389 |
| <i>CHAMP: Efficient Annotation and Consolidation of Cluster Hierarchies</i><br>Arie Cattan, Tom Hope, Doug Downey, Roy Bar-Haim, Lilach Eden, Yoav Kantor and Ido Dagan   | 403 |
| <i>Prompt2Model: Generating Deployable Models from Natural Language Instructions</i><br>Vijay Viswanathan, Chenyang Zhao, Amanda Bertsch, Tongshuang Wu and Graham Neubig   | 413 |
| <i>NewsSense: Reference-free Verification via Cross-document Comparison</i><br>Jeremiah Milbauer, Ziqi Ding, Zhijin Wu and Tongshuang Wu .....  | 422 |
| <i>NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails</i><br>Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien and Jonathan Cohen .....   | 431 |
| <i>LM-Polygraph: Uncertainty Estimation for Language Models</i><br>Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin and Artem Shelmanov .....   | 446 |
| <i>Descriptive Knowledge Graph in Biomedical Domain</i><br>Kerui Zhu, Jie Huang and Kevin Chen-Chuan Chang .....  | 462 |
| <i>Prompterator: Iterate Efficiently towards More Effective Prompts</i><br>Samuel Sučík, Daniel Skala, Andrej Švec, Peter Hraška and Marek Šuppa .....  | 471 |
| <i>ZhuJiu: A Multi-dimensional, Multi-faceted Chinese Benchmark for Large Language Models</i><br>Baoli Zhang, Haining Xie, Pengfan Du, Junhao Chen, Pengfei Cao, Yubo Chen, Shengping Liu, Kang Liu and Jun Zhao .....  | 479 |
| <i>PaperMage: A Unified Toolkit for Processing, Representing, and Manipulating Visually-Rich Scientific Documents</i><br>Kyle Lo, Zejiang Shen, Benjamin Newman, Joseph Chee Chang, Russell Authur, Erin Bransom, Stefan Candra, Yoganand Chandrasekhar, Regan Huff, Bailey Kuehl, Amanpreet Singh, Chris Wilhelm, Angele Zamarron, Marti A. Hearst, Daniel Weld, Doug Downey and Luca Soldaini ..... | 495 |

|  |     |
|--|-----|
| <i>OmniEvent: A Comprehensive, Fair, and Easy-to-Use Toolkit for Event Understanding</i>   |     |
| Hao Peng, Xiaozhi Wang, Feng Yao, Zimu Wang, Chuzhao Zhu, Kaisheng Zeng, Lei Hou and Juanzi Li .....   | 508 |
| <i>CocoSciSum: A Scientific Summarization Toolkit with Compositional Controllability</i>   |     |
| Yixi Ding, Yanxia Qin, Qian Liu and Min-Yen Kan .....  | 518 |
| <i>CoLLiE: Collaborative Training of Large Language Models in an Efficient Way</i>   |     |
| Kai Lv, Shuo Zhang, Tianle Gu, Shuhao Xing, Jiawei Hong, Keyu Chen, Xiaoran Liu, Yuqing Yang, Honglin Guo, Tengxiao Liu, Yu Sun, Qipeng Guo, Hang Yan and Xipeng Qiu .....     | 527 |
| <i>Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding</i>   |     |
| Hang Zhang, Xin Li and Lidong Bing .....   | 543 |
| <i>SummHelper: Collaborative Human-Computer Summarization</i>  |     |
| Aviv Slobodkin, Niv Nachum, Shmuel Amar, Ori Shapira and Ido Dagan .....   | 554 |
| <i>ModelScope-Agent: Building Your Customizable Agent System with Open-source Large Language Models</i>  |     |
| Chenliang Li, He Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, Hongzhu Shi, Ji Zhang, Fei Huang and Jingren Zhou | 566 |
| <i>EfficientOCR: An Extensible, Open-Source Package for Efficiently Digitizing World Knowledge</i>   |     |
| Tom Bryan, Jacob Carlson, Abhishek Arora and Melissa Dell .....  | 579 |

# FABRICATOR: An Open Source Toolkit for Generating Labeled Training Data with Teacher LLMs

Jonas Golde<sup>1</sup>, Patrick Haller<sup>1</sup>, Felix Hamborg<sup>1</sup>, Julian Risch<sup>2</sup>, Alan Akbik<sup>1</sup>

<sup>1</sup> Humboldt University of Berlin

<sup>2</sup> deepset GmbH

{jonas.golde, patrick.haller.1, felix.hamborg, alan.akbik}@hu-berlin.de  
julian.risch@deepset.ai

## Abstract

Most NLP tasks are modeled as supervised learning and thus require labeled training data to train effective models. However, manually producing such data at sufficient quality and quantity is known to be costly and time-intensive. Current research addresses this bottleneck by exploring a novel paradigm called *zero-shot learning via dataset generation*. Here, a powerful LLM is prompted with a task description to generate labeled data that can be used to train a downstream NLP model. For instance, an LLM might be prompted to “generate 500 movie reviews with positive overall sentiment, and another 500 with negative sentiment.” The generated data could then be used to train a binary sentiment classifier, effectively leveraging an LLM as a teacher to a smaller student model. With this demo, we introduce FABRICATOR, an open-source Python toolkit for dataset generation. FABRICATOR implements common dataset generation workflows, supports a wide range of downstream NLP tasks (such as text classification, question answering, and entity recognition), and is integrated with well-known libraries to facilitate quick experimentation. With FABRICATOR, we aim to support researchers in conducting reproducible dataset generation experiments using LLMs and help practitioners apply this approach to train models for downstream tasks.

## 1 Introduction

In recent years, natural language processing (NLP) has witnessed remarkable progress due to the introduction of pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019; Conneau and Lample, 2019; He et al., 2021). These PLMs are typically fine-tuned on large human-annotated datasets, resulting in state-of-the-art performance in tasks such as text classification, token classification, and question answering. However, real-world

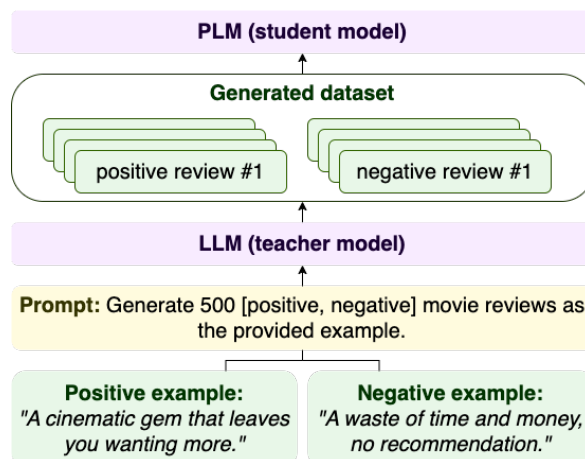


Figure 1: The process of *learning via dataset generation*. A teacher model (LLM) is prompted to generate 500 movie reviews for each sentiment (positive, negative). A smaller student PLM is trained on the generated dataset.

applications of this approach face the bottleneck that sufficient amounts of human-annotated data are often unavailable and too costly to produce manually, especially when domain expertise is required.

**Dataset generation with teacher LLMs.** Recently, a paradigm called *zero-shot learning via dataset generation* (Meng et al., 2022; Ye et al., 2022a,b) has emerged, potentially obviating the need for human-annotated data. This approach leverages the generation capability of large language models (LLMs) to create class-conditioned texts guided by label-descriptive prompts and, optionally, *few-shot* examples of instances of the desired classes. The generated dataset is then used to train a smaller student PLM.

Refer to Figure 1 for an illustration of this process: In this example, an LLM is instructed to write 500 positive and 500 negative movie reviews. To guide the process, we include an example of a positive and negative review in the prompt. With this

prompt and 1-shot example, we generate a dataset of 1,000 movie reviews labeled with binary sentiment. This dataset is used to train a student model to perform binary sentiment analysis.

**Limitations.** However, despite the conceptual simplicity of using LLMs to generate training data, many open questions remain regarding the specifics and ultimate potential of this approach. Questions include: (1) How to best prompt the LLM and whether to include examples in the prompt, (2) For which downstream NLP task families and specific tasks this approach is effective, and (3) Whether it is better to generate large amounts of training data or focus on smaller, high-quality generation efforts. While various current works are investigating these questions for specific tasks, we find that, at present, no open-source library specifically supports research on dataset generation with LLMs.

**Contributions.** To close this gap, we present FABRICATOR, an open-source Python library for dataset generation with LLMs. Our main goals are to facilitate experimentation, enable the application of dataset generation to specific downstream tasks, and encourage the reproducibility of experiments.

FABRICATOR modularizes the dataset generation process and provides a simple interface to facilitate experimentation: Users may choose which LLM to use, define prompts and label definitions, and leverage existing NLP datasets for few-shot examples and NLP task definitions. Our library includes an integration into HuggingFace’s DATASETS library (Lhoest et al., 2021), allowing users to easily share generated datasets and use them for training NLP models. We provide examples for various NLP task families, including text classification, textual entailment, question answering, and entity recognition. In this paper:

- We introduce the FABRICATOR library, give an overview of core concepts and usage workflows (Section 2).
- We present a set of example experiments in which FABRICATOR is used to create datasets for various text classification, question answering, and textual entailment tasks (Section 3).

We publish the code on GitHub<sup>1</sup> under the Apache 2 license.

<sup>1</sup><https://github.com/flairNLP/fabricator>

## 2 FABRICATOR

We first give a high-level overview of supported generation workflows in FABRICATOR (Section 2.1), discuss the main classes and concepts (Section 2.2), and walk through an example use case and script (Section 2.3).

### 2.1 Generation Workflows

Depending on the downstream task, researchers may have one of three data generation targets we support in FABRICATOR:

**1. Generate unlabeled data.** The first generation target is to *produce unlabeled data*. For instance, during the development of a question answering system, we might require a corpus of example questions or a corpus of texts on a particular topic. For this scenario, users provide a prompt  $w$  (such as “Generate a text in the domain of history that contains facts someone can ask questions about.”), and the auto-regressive LLM  $G_\theta$  generates appropriate text  $x^g$ .

**2. Generate label-conditioned data.** The second generation target is generating data belonging to a pre-defined class, such as classification tasks. The LLM generates a text  $x^g$  corresponding to a specific label  $y$  from a set of labels.

As discussed in the introduction, one example is to generate training data for a binary sentiment classifier. To achieve this, one must define a set of labels ( $y = \{positive, negative\}$ ) and a prompt  $w_y$  such as “Generate a  $\langle y \rangle$  movie review:.” The generated sequence  $x^g$  will be paired with the label  $y$  to form a training pair  $(x^g, y)$  for fine-tuning.

**3. Annotate unlabeled data.** The third generation target holds if an unlabeled text dataset for a domain is already available and only training labels are missing. For instance, a corpus of movie reviews might already be available, but sentiment labels are missing.

In FABRICATOR, researchers can add labels to an existing corpus by extending prompt  $w$  with fixed label options  $y$  to form  $w_y$  like “Annotate the movie review either as: *positive, negative*.” The generated label  $y$  is then paired with the unlabeled data point  $x^u$  to form a data pair  $(x^u, y)$ .

The generation targets defined above will be executed multiple times to generate a corpus of a specified size. The prompt may also be extended to

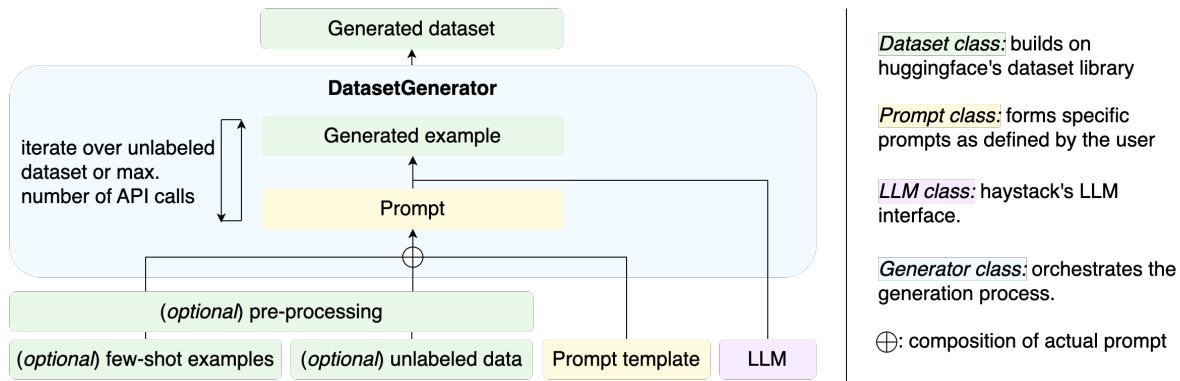


Figure 2: With FABRICATOR, the generation process involves a prompt template that creates the final prompt using all provided arguments. The generator class creates training examples until the maximum number of prompt calls is reached, or the unlabeled dataset is fully annotated. Ultimately, the generator class produces a HuggingFace Dataset instance.

include few-shot examples of each class, as shown in Figure 1. The prompt can also handle multiple inputs (for example, for tasks like textual similarity) using pre-defined interfaces in FABRICATOR. In all cases, the correct prompt is composed and executed in our backend.

## 2.2 Classes and Concepts

As Figure 2 illustrates, the key module in our approach is the `DatasetGenerator` class, which acts as an orchestrator between the LLM (`PromptNode`), the prompt (`BasePrompt`), and optionally, the few-shot examples and unlabeled datasets.

The `generate()` function within the `DatasetGenerator` class converts the `BasePrompt` and the provided few-shot and unlabeled data into a processable prompt for the LLM. The method offers various arguments to steer the generation process. Users can specify parameters like the maximum number of API calls, the sampling strategy of few-shot examples (uniform vs. stratified), or the number of few-shot examples to use in a single prompt. Our repository contains documentation with details on all available customization options.

### 2.2.1 HuggingFace Interoperability through Dataset Class

FABRICATOR operates on the `Dataset` class from HuggingFace’s `DATASETS` library. By default, `generate()` produces the generated data as a `Dataset` instance. This allows generated datasets to be directly used in existing training scripts of the `TRANSFORMERS` library (Wolf et al., 2020) and to be shared among researchers via the Huggingface dataset hub.

An existing dataset may also be used as input to the `generate()` method. Since the `DATASETS` library supports a wide range of standard benchmarks and their formats, existing datasets can be easily loaded and used as input. For instance, in some generation workflows, we would like to add labels to an existing corpus or use instances as few-shot examples within a prompt.

### 2.2.2 Prompt Class

Prompting is crucial when operating on large language models as it guides the auto-regressive generation process. While in the simplest case, a prompt is a single textual string, we find that many scenarios require more complex prompts and customization options. For instance, when including few-shot examples in a prompt, questions include how many examples to include in each prompt and how these are sampled (uniform vs. stratified) from available few-shot data across different prompt calls. Similarly, the complexity increases for tasks such as textual entailment (requiring multiple inputs) and entity recognition (potentially requiring transformation of token-level BIOES tags into span-level prompting queries).

To address these challenges, FABRICATOR introduces a simple yet powerful `BasePrompt` class that offers clear interfaces for customizing prompts for various dataset generation tasks. The interface includes attributes to specify pre-defined label options for label-conditioned generation, and support for having few-shot examples or unlabeled datasets by selecting the relevant columns for generation and few-shot information in the prompt.

Since the prompt class directly operates on the dataset columns, FABRICATOR enables a sophis-

```

1 import os
2 from datasets import load_dataset
3 from haystack.nodes import PromptNode
4 from fabricator import DatasetGenerator, BasePrompt
5
6 dataset = load_dataset("processed_fewshot_imdb", split="train")
7
8 prompt = BasePrompt(
9     task_description="Generate a {} movie review.",
10    label_options=["positive", "negative"],
11    generate_data_for_column="text",
12 )
13
14 prompt_node = PromptNode(
15    model_name_or_path="gpt-3.5-turbo",
16    api_key=os.environ.get("OPENAI_API_KEY"),
17    max_length=100,
18 )
19
20 generator = DatasetGenerator(prompt_node)
21 generated_dataset = generator.generate(
22    prompt_template=prompt,
23    fewshot_dataset=dataset,
24    fewshot_sampling_strategy="uniform",
25    fewshot_examples_per_class=1,
26    fewshot_sampling_column="label",
27 )
28 generated_dataset.push_to_hub("generated-movie-reviews")

```

Listing 1: A script that uses FABRICATOR and generates additional movie reviews based on few-shot examples.

licated and flexible prompt design. To illustrate, when performing a textual similarity task, the user can specify the first sentence and the label as the few-shot information and prompt the LLM to generate a second sentence corresponding to the given sentence and label.

### 2.2.3 LLMs

The LLM interface must be stable and ideally compatible with models hosted as APIs or self-hosted LLMs. We leverage the HAYSTACK<sup>2</sup> framework (Pietsch et al., 2019), specifically the PromptNode class, for interactions with LLMs. The PromptNode implementation allows users to select and use LLMs from various model providers, including HuggingFace, OpenAI, Azure, Anthropic, and Cohere.

## 2.3 Example Script

In Listing 1, we introduce an example script in which FABRICATOR is used to generate additional movie reviews for training a binary sentiment classification model (refer to generation workflow 2 as defined in Section 2.1). To implement this, we define:

- a pre-processed few-shot dataset (dataset, line 6) having labels in natural language form (e.g., 0 becomes “negative”). These examples are used to augment the generation prompt,
- a prompt template (prompt, line 8) specifying the instruction to the LLM,
- an LLM to use as teacher model (prompt\_node, line 14),
- a DatasetGenerator to execute the generation process with all parameters (generator, line 20).

The prompt is configured in the constructor of the BasePrompt class (lines 8-12): We set a task\_description with a placeholder for label\_options that we provide as a separate argument. We also specify for which column in the loaded dataset to predict labels.

We then define a teacher LLM (lines 14-18) and pass datasets, prompt, and LLM to the DatasetGenerator orchestrator class (lines 20-27). Here, we specify a few-shot strategy to sample one label from the “label” column uniformly during generation. We do so to generate either a positive or a negative review. Upon completion, the generate function returns the annotated Dataset instance.

<sup>2</sup><https://github.com/deepset-ai/Haystack>



| Dataset | Labels    | # Training examples |                   |                   |                   |
|---------|-----------|---------------------|-------------------|-------------------|-------------------|
|         |           | 50                  | 500               | 1k                | all (max. 10k)    |
| IMDB    | Gold      | 37.6 ± 35.8         | 88.5 ± 0.8        | 90.0 ± 0.4        | 93.0 ± 0.2        |
|         | Generated | <b>53.8 ± 11.5</b>  | <b>88.8 ± 0.6</b> | <b>90.2 ± 0.4</b> | <b>92.0 ± 0.1</b> |
| MRPC    | Gold      | 66.6 ± 0.8          | 73.0 ± 1.3        | 75.2 ± 1.1        | 83.9 ± 0.2        |
|         | Generated | <b>68.4 ± 0.8</b>   | <b>72.1 ± 1.0</b> | 72.4 ± 1.2        | 75.8 ± 0.7        |
| SNLI    | Gold      | 38.5 ± 2.5          | 64.7 ± 0.9        | 71.3 ± 0.7        | 82.1 ± 0.4        |
|         | Generated | <b>42.2 ± 2.4</b>   | 54.8 ± 1.0        | 56.1 ± 1.1        | 63.1 ± 0.7        |
| TREC-6  | Gold      | 50.4 ± 7.6          | 93.6 ± 0.6        | 94.9 ± 1.1        | 97.5 ± 0.4        |
|         | Generated | 39.8 ± 4.5          | 79.3 ± 2.2        | 80.8 ± 3.0        | 82.4 ± 1.1        |
| SQuAD   | Gold      | -                   | -                 | 39.1 ± 4.9        | 68.8 ± 0.5        |
|         | Generated | -                   | -                 | <b>46.8 ± 1.1</b> | 52.5 ± 0.3        |

Table 1: Results on re-annotation experiments using 2 few-shot examples per prompt (uniformly sampled from 6 few-shot examples per class). We report accuracy except for SQuAD, where we report F1, and highlight bold those experiments where generated data yielded similar scores as human-annotated data. We observe that GPT-3.5 is not able to annotate on human-level performance except for simple classification tasks such as IMDB.

### 3 Experiments

To illustrate how FABRICATOR could be used in research, we conduct an exploratory evaluation of two scenarios: (1) how models trained on generated datasets compare to models trained on human-annotated datasets, and (2) whether few-shot examples in the prompt improve generated datasets.

To do so, we train smaller PLMs on generated datasets and evaluate them on the human-labeled test split of the respective benchmark. For question answering, we fine-tune a roberta-base PLM (Liu et al., 2019). For all other tasks, we fine-tune a bert-base-uncased PLM (Devlin et al., 2019). The hyperparameters are listed in Appendix A.2. We report the score and standard deviation averaged over 5 random seeds for each experiment.

#### 3.1 Experiment 1: Comparison of Generated and Human-Annotated Datasets

We re-annotate existing benchmark datasets with generated labels in the first experiment. This experiment aims to measure the difference in accuracy of downstream task models trained on human-annotated data compared to models trained on generated data. We evaluate text classification, textual similarity, and extractive question answering tasks. **Experimental setup.** We conduct this evaluation on 5 datasets spanning 3 NLP tasks: We use IMDB (Maas et al., 2011), a binary sentiment classification benchmark, and TREC-6 (Li and Roth, 2002),

a 6-class question type categorization dataset to evaluate text classification tasks. We use the 2-class MRPC (Dolan and Brockett, 2005) and the 3-class SNLI (Bowman et al., 2015) datasets to evaluate textual similarity tasks. Finally, we use SQuAD-v2 (Rajpurkar et al., 2016) to evaluate extractive question answering. We use generation prompts augmented by 2 examples per prompt sampled from 6 possible few-shot examples per class. **Results (Table 1).** For all datasets, we compare a generated dataset of 50, 500, 1k and the full dataset (limited to 10k if it is larger) to gold-annotated data of the same size. For question answering, models need to be trained on at least 1k to obtain representative results, so we do not report scores for 50 or 500 examples for SQuAD.

We find that for simple tasks such as binary sentiment classification (IMDB), models trained on the annotations by LLMs achieve similar accuracy on the gold-labeled test split ( $\downarrow 1.0$  pp. in accuracy with 10k training examples). However, as the complexity of datasets increases (text classification with more classes and extractive question answering), we observe that the performance of models trained on LLM-annotated datasets falls short ( $\downarrow 19.0$  pp. for SNLI and  $\downarrow 16.3$  pp. for SQuAD, with 10k training examples).

These performance gaps indicate that the usefulness of LLMs as teacher models depends on the specific task. In the next section, we present an experiment that explores how to close this gap by using additional few-shot examples.

| Dataset | # few-shot examples<br>per class | # examples per class used in prompt |            |                   |                   |                   |
|---------|----------------------------------|-------------------------------------|------------|-------------------|-------------------|-------------------|
|         |                                  | 0                                   | 1          | 2                 | 3                 | 4                 |
| TREC-6  | 0                                | 45.5 ± 2.3                          | -          | -                 | -                 | -                 |
|         | 2                                | -                                   | 70.0 ± 1.6 | 65.5 ± 0.9        | -                 | -                 |
|         | 4                                | -                                   | 79.5 ± 1.1 | 71.1 ± 2.0        | <b>86.6 ± 0.6</b> | 69.8 ± 1.5        |
|         | 8                                | -                                   | 76.1 ± 1.9 | <b>79.5 ± 1.3</b> | <b>81.0 ± 1.8</b> | <b>87.4 ± 0.6</b> |
|         | 16                               | -                                   | 72.7 ± 2.1 | 78.1 ± 1.9        | <b>81.0 ± 2.4</b> | 74.2 ± 1.4        |

Table 2: Results on 500 annotated TREC-6 examples using varying amounts of few-shot examples. We sweep over the number of few-shot examples and the number of few-shot examples used in the actual prompt. We highlight bold where increasing few-shot examples improves over the 79.3 TREC-6 score of Experiment 1 (Table 1).

### 3.2 Experiment 2: Impact of Few-Shot Examples

In the second example experiment, we re-annotate TREC-6 using a varying number of few-shot examples. This experiment aims to determine whether adding few-shot examples for each class improves dataset generation with FABRICATOR. We investigate two variables: (1) The total number of available few-shot examples per class and (2) the actual number of few-shot examples included per prompt. For instance, there might be 8 few-shot examples available in total, but only 3 are randomly sampled to be included in each prompt call.

**Results (Table 2).** We note a generally positive trend in that increasing the number of available few-shot examples (column *# few-shot examples per class*) and increasing the number of examples used in each prompt (column *# examples per class used in prompt*) improves model performance. In particular, we find many settings that outperform the numbers of our previous experiment (where we sampled 2 examples per prompt out of a total of 6 possible examples), highlighted bold in Table 2.

However, we also find that improvements become uneven when *# examples per class used in prompt* is increased above 3, indicating prompts should not be overloaded with too many examples.

## 4 Related Work

Significant progress has been achieved in enhancing dataset generation with teacher LLMs (Schick and Schütze, 2021b; Meng et al., 2022; Ye et al., 2022a; Bonifacio et al., 2022; Peng et al., 2023; Meng et al., 2023), effectively selecting few-shot examples (Liu et al., 2022; Gunasekar et al., 2023) and assessing the quality of datasets produced by LLMs (Gilardi et al., 2023; Chen et al., 2023).

However, we note a lack of accessible frameworks that facilitate straightforward and reproducible dataset generation using teacher LLMs. While existing open-source toolkits like OpenPrompt (Ding et al., 2022) partially extend to dataset generation scenarios, our approach stands apart by having lightweight, dedicated interfaces for the introduced generation tasks, supporting a wide range of LLMs using haystack, and integrating with HuggingFace DATASETS for easy evaluation.

Prompt-based learning (Liu et al., 2021; Gao et al., 2021; Schick and Schütze, 2021a; Le Scao and Rush, 2021) is another line of research that has proven useful in improving downstream tasks in zero- and few-shot settings by leveraging LLMs’ pre-training objectives (Brown et al., 2020; Ouyang et al., 2022; Zhang et al., 2022; Scao et al., 2023; Touvron et al., 2023). However, the availability of training data in low-resource scenarios is still crucial (Perez et al., 2021; Sahu et al., 2022). Therefore, our method also seeks to fill this gap by providing a comprehensive and easily reproducible dataset generation toolkit.

## 5 Conclusion

We introduced FABRICATOR, a user-friendly library for dataset generation utilizing LLMs. With FABRICATOR, researchers access a highly customizable interface that enables efficient research on zero-shot and few-shot learning via dataset generation. Further, we implemented various baselines using generated datasets to illustrate potential applications of our repository and plan to support further downstream tasks in the future. We believe that FABRICATOR will be a valuable tool for the NLP community, facilitating advancements in dataset generation and fostering research in various natural language processing domains.

## Limitations

While our paper aims to address dataset creation for a wide range of downstream tasks, it is important to acknowledge certain limitations in our study. Firstly, during our repository’s evaluation phase, we could only test and assess a subset of tasks due to resource and time constraints. Our evaluation may only cover a portion of the tasks researchers and practitioners commonly encounter in their work. Future work must expand the evaluation to include a broader range of tasks to provide a more comprehensive understanding of the repository’s effectiveness.

Additionally, despite our best efforts in designing the repository layout to be versatile and adaptable, there might be specific tasks or domains where our repository’s structure or features may not be directly applicable. We acknowledge that the landscape of downstream tasks is diverse and constantly evolving, which may require tailored approaches or extensions to our existing framework. Further, we aim to include existing research targeting high-quality dataset generation (e.g., [Ye et al. \(2022b\)](#)) and conduct our own research on quality and diversity metrics to steer the generation process. We encourage open-source contributions and active engagement from the community to address these limitations. By involving a more comprehensive range of perspectives and expertise, we aim to consistently improve the repository and enhance its suitability for various task requirements.

Furthermore, while we have endeavored to provide thorough documentation and guidelines within the repository, there is always a possibility of overlooked issues or unforeseen challenges that may arise during dataset creation.

## Ethics Statement

While large language models have shown remarkable advancements in natural language understanding and generation, their capabilities also raise important ethical considerations. One prominent concern is the potential for hallucination, where the models may generate false or misleading information. This aspect can have serious implications, especially when datasets are created for critical domains such as medicine, law, or journalism. It is crucial to exercise caution and verify the accuracy and reliability of outputs generated by our repository, particularly when making decisions that have real-world consequences.

Another ethical concern is the presence of biases in language models, which can perpetuate and amplify societal prejudices and inequalities. These biases can arise from biased training data ([Haller et al., 2023](#)) or biased patterns in human-generated text that the models learn from. Since our repository is in an early stage, we emphasize to carefully inspect created datasets to identify and rectify biases that may be present.

To ensure a responsible dataset creation process, it is essential to engage in thorough data validation, including identifying and addressing potential biases, checking data sources for reliability and credibility, and involving diverse perspectives in dataset collection and annotation processes. Moreover, continuous monitoring and auditing of the models’ outputs and performance can help identify and rectify any ethical concerns arising during deployment.

## Acknowledgements

We thank all reviewers for their valuable comments. Jonas Golde is supported by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) as part of the project ENA (KK5148001LB0). Alan Akbik and Patrick Haller are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Emmy Noether grant “Eidetic Representations of Natural Language” (project number 448414230). Alan Akbik is furthermore supported under Germany’s Excellence Strategy “Science of Intelligence” (EXC 2002/1, project number 390523135). Felix Hamborg is supported by the WIN program of the Heidelberg Academy of Sciences and Humanities, financed by the Ministry of Science, Research and Arts of the State of Baden-Württemberg, Germany.

## References

- Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. [Inpars: Unsupervised dataset generation for information retrieval](#). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’22*, page 2387–2392, New York, NY, USA. Association for Computing Machinery.
- 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.
- Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2023. [An empirical survey of data augmentation for limited data learning in NLP](#). *Transactions of the Association for Computational Linguistics*, 11:191–211.
- Alexis Conneau and Guillaume Lample. 2019. [Cross-lingual language model pretraining](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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.
- William B. Dolan and Chris Brockett. 2005. [Automatically constructing a corpus of sentential paraphrases](#). In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. [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.
- Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. [ChatGPT outperforms crowd workers for text-annotation tasks](#). *Proceedings of the National Academy of Sciences*, 120(30).
- Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. [Textbooks are all you need](#).
- Patrick Haller, Ansar Aynedinov, and Alan Akbik. 2023. [Opiniongpt: Modelling explicit biases in instruction-tuned llms](#).
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [Deberta: Decoding-enhanced bert with disentangled attention](#). In *International Conference on Learning Representations*.
- Teven Le Scao and Alexander Rush. 2021. [How many data points is a prompt worth?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. 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, Gungjan 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.
- Xin Li and Dan Roth. 2002. [Learning question classifiers](#). In *COLING 2002: The 19th International Conference on Computational Linguistics*.
- 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.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. [Gpt understands, too](#).
- 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](#).

- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. [Generating training data with language models: Towards zero-shot language understanding](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 462–477. Curran Associates, Inc.
- Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023. [Tuning language models as training data generators for augmentation-enhanced few-shot learning](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24457–24477. PMLR.
- 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.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. [Instruction tuning with gpt-4](#).
- Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. [True few-shot learning with language models](#). In *Advances in Neural Information Processing Systems*, volume 34, pages 11054–11070. Curran Associates, Inc.
- Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. 2019. [Haystack: the end-to-end NLP framework for pragmatic builders](#).
- 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.
- Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. 2022. [Data augmentation for intent classification with off-the-shelf large language models](#). In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland. Association for Computational Linguistics.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellice Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, and Matthias Gallé et al. 2023. [Bloom: A 176b-parameter open-access multilingual language model](#).
- Timo Schick and Hinrich Schütze. 2021a. [Exploiting cloze-questions for few-shot text classification and natural language inference](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. 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.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. [Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition](#). In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- 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](#).
- 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.
- Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022a. [ZeroGen: Efficient zero-shot learning via dataset generation](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11653–11669, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2022b. [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.

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 Appendix

### A.1 Screencast

A screencast about the FABRICATOR framework can be found on [Vimeo](#).

### A.2 Hyperparameters for Experiments

We used AdamW (Loshchilov and Hutter, 2019) as our optimizer with a batch size of 16. Further, we used a linear warm-up for 10% of the optimization steps. We fine-tune roberta-base for question answering with a learning rate of  $1e^{-5}$  for two epochs without early stopping. For the bert-base-uncased PLM, we fine-tune using a learning rate of  $2e^{-5}$  for either 5 (if training data has more than 1000 examples), 10 (if training dataset has at least 500 but less than 1001 examples) or 20 epochs (if training data is less than 501 examples). Further, across all experiments, we use 10% of the data as a validation split for model selection.

### A.3 Generate Label-Conditioned Training Data

This experiment used label-conditioned generation to create new data for the TREC dataset containing six classes. To achieve this, we sampled a small few-shot dataset from the existing training split, consisting of 8 examples per class. During generation, for each label  $y$ , we included three uniformly sampled few-shot examples associated with that label. We generated 10k data pairs  $(x^g, y)$  and used them for fine-tuning. It is important to note that the gold-labeled dataset contains only around 3k examples. Thus the column “all” refers either to the 10k examples generated with GPT or to the ~3k gold-labeled examples. The experimental setup is identical to Section 3.

The results are depicted in Table 3. We observe significant performance drops compared to

the re-annotation experiments for TREC from Section 3.1. For instance, using 10k generated examples achieves a performance level similar to using 50 human-annotated examples (compare to Table 1). However, we note that we performed no prompt optimization techniques or hyperparameter searches in all experiments. Additionally, we generated a uniform distribution of classes, while the gold-labeled dataset is skewed towards certain categories. It is worth mentioning that this class distribution information may not be available in real-world few-shot settings.

### A.4 Impact of Few-Shot Examples on Label-Conditioned Generation

In this experiment, we generated 500 label-conditioned data pairs for the TREC dataset, following the approach described in Section 3.2. We conducted a sweeping analysis over two factors: the total number of few-shot examples per class and the number of few-shot examples included in the actual prompt.

The results are depicted in Table 4. Our findings show that including even a small number of few-shot examples ( $< 4$ ) yields better results compared to generating without any few-shot examples. Moreover, when we used at least four examples per class, we observed significant improvements in the generation results, from 30.2 to 54.8 in accuracy ( $\uparrow 24.6$  pp. in accuracy). Additionally, using more examples in a distinct prompt slightly improved the model performance. We encountered one outlier when using 16 examples per class and including five examples in the prompt for generation, which resulted in lower performance than sampling from 8 few-shot examples per prompt. It is important to note that during this experiment, we did not adjust any hyper-parameters of the LLM for generation, such as temperature or top-k sampling.

### A.5 Instruction-tuning open-source models

In this experiment, we compare the annotation performance of OpenAI’s GPT-3.5 with an instruction-tuned open-source LLaMA model. To conduct this evaluation, we choose the token classification task on the CoNLL-03 dataset (Tjong Kim Sang and De Meulder, 2003), which generates one label for each token in the input, making it a structured task.

The results are shown in Table 5. We observe that using the dataset as-is results in often unusable annotation outputs, primarily due to imprecise formatting. To address this, we convert the token-level

| Dataset | Data      | # Training examples |            |            |            |
|---------|-----------|---------------------|------------|------------|------------|
|         |           | 50                  | 500        | 1000       | all        |
| TREC-6  | Gold      | 42.7 ± 9.6          | 93.8 ± 0.3 | 95.1 ± 0.6 | 97.1 ± 0.3 |
|         | Generated | 27.5 ± 11.0         | 56.2 ± 3.3 | 57.9 ± 1.6 | 62.6 ± 3.4 |

Table 3: Results on TREC-6 with generated questions by GPT-3.5 using 3 few-shot examples (uniformly sampled from 8 possible few-shot examples per class). We observe that the generation performance is worse compared to an equally sized human-annotated dataset. However, the performance increases with the number of examples generated.

| Dataset | # few-shot examples<br>per class | # examples per class used in prompt |            |            |            |            |
|---------|----------------------------------|-------------------------------------|------------|------------|------------|------------|
|         |                                  | 0                                   | 2          | 3          | 4          | 5          |
| TREC-6  | 0                                | 30.2 ± 0.6                          | -          | -          | -          | -          |
|         | 2                                | -                                   | 43.0 ± 3.7 | -          | -          | -          |
|         | 4                                | -                                   | 56.0 ± 0.5 | 56.3 ± 2.4 | 58.3 ± 2.2 | -          |
|         | 8                                | -                                   | 52.8 ± 1.5 | 58.8 ± 1.0 | 58.2 ± 1.0 | 64.0 ± 2.0 |
|         | 16                               | -                                   | 58.3 ± 0.8 | 59.8 ± 2.5 | 58.7 ± 1.1 | 54.8 ± 1.5 |

Table 4: Results on 500 generated TREC-6 examples with different sizes of few-shot examples and number of few-shot examples included in the prompt. We observe that more few-shot examples result in better performance on the gold annotated test split.

| Model                   | Acc. | (micro) F1 |
|-------------------------|------|------------|
| LLaMAv2 + Instr. Tuning | 92.4 | 60.0       |
| GPT-3.5*                | 88.4 | 52.5       |

Table 5: Comparison of instruction-tuned LLaMA models with 3-shot GPT-3.5 based on the training split of CoNLL-03. We report accuracy and span-level F1 score the annotation on the validation split. \*: We convert tag sequences to spans in order to prompt the LLM with strings rather than sequence. However, 38% of the validation split annotations have different lengths after tokenization which have been filtered out for a fair comparison.

labels into spans and prompt the LLM to extract all named entities for the relevant categories. We then transform the found entities into token-level tags by searching for the annotations as substrings of the input text. We compare the performance of this approach with a instruction-tuned LLaMA model on the entire training split of CoNLL-03 by letting both LLMs annotate the validation set.

Unlike the previous evaluation, we did not train and evaluate a smaller PLM on the gold-labeled test set. Instead, we assess the performance between the gold-annotated validation split and the annotations made by the LLM. Our findings indi-

cate that the annotation quality of instruction-tuned LLMs can significantly improve over OpenAI’s GPT, as evident from the higher F1 score. This finding suggests that instruction-tuned models for dataset generation have the potential to facilitate the generation process for complex downstream tasks in future research endeavors.

# End-to-End Evaluation for Low-Latency Simultaneous Speech Translation

Christian Huber<sup>1</sup>, Tu Anh Dinh<sup>1</sup>, Carlos Mullov<sup>1</sup>, Ngoc Quan Pham<sup>1</sup>,  
Thai Binh Nguyen<sup>1</sup>, Fabian Retkowski<sup>1</sup>, Stefan Constantin<sup>1</sup>, Enes Yavuz Ugan<sup>1</sup>,  
Danni Liu<sup>1</sup>, Zhaolin Li<sup>1</sup>, Sai Koneru<sup>1</sup>, Jan Niehues<sup>1</sup> and Alexander Waibel<sup>1,2</sup>

<sup>1</sup>Karlsruhe Institute of Technology, Karlsruhe, Germany

firstname.lastname@kit.edu

<sup>2</sup>Carnegie Mellon University, Pittsburgh PA, USA

alexander.waibel@cmu.edu

## Abstract

The challenge of low-latency speech translation has recently draw significant interest in the research community as shown by several publications and shared tasks. Therefore, it is essential to evaluate these different approaches in realistic scenarios. However, currently only specific aspects of the systems are evaluated and often it is not possible to compare different approaches.

In this work, we propose the first framework to perform and evaluate the various aspects of low-latency speech translation under realistic conditions. The evaluation is carried out in an end-to-end fashion. This includes the segmentation of the audio as well as the run-time of the different components.

Secondly, we compare different approaches to low-latency speech translation using this framework. We evaluate models with the option to revise the output as well as methods with fixed output. Furthermore, we directly compare state-of-the-art cascaded as well as end-to-end systems. Finally, the framework allows to automatically evaluate the translation quality as well as latency and also provides a web interface to show the low-latency model outputs to the user.

## 1 Introduction

In many applications scenarios for speech translation, the quality of the translations is not the only important metric, but it is also essential to provide the translation with a low latency. This is for example the case in translations of presentations or meetings. Therefore, we observe an increasing interest in the field of low-latency speech translations, as shown by numerous published techniques and the organization of a dedicated shared task as part of the International Conference on Spoken Language Translations (IWSLT) (Agrawal et al., 2023).

In order to enable further progress in the field as well as a wide adoption of the technique a framework to evaluate different approaches is essential.

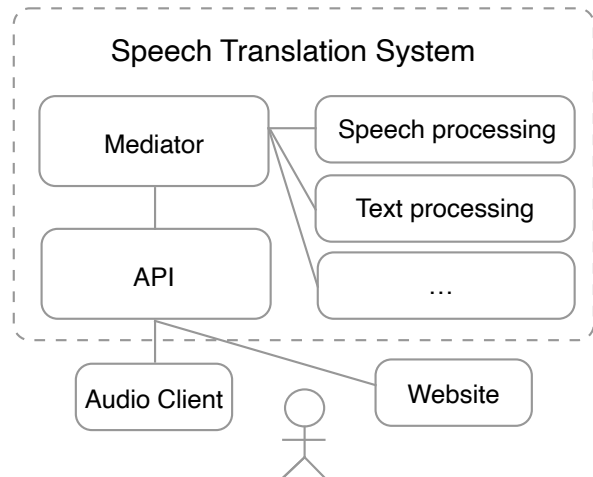


Figure 1: Framework overview

However, the current evaluation only considers a limited number of aspects or techniques. In contrast, for an overall evaluation of different architectures (end-to-end and cascaded) and presentation style (revision and fixed) a general evaluation framework is needed. This should also consider the computational latency as well as the ability to process several sessions in parallel.

Motivated by this, we present a new framework to apply and evaluate low-latency, simultaneous speech translation. Thereby we focus on a framework that can evaluate the different approaches in as realistic conditions as possible. The system is able to simulate different load conditions as well as compare systems using different design choices. Finally, we also provide a web interface<sup>1</sup> to present the low-latency model outputs to the user.

The main contributions of our paper are:

- A framework<sup>2</sup> for low-latency speech translation with dynamic latency adjustment
- An evaluation setup that allows for assessing the quality and latency of a low-latency scenario in an end-to-end fashion

<sup>1</sup><https://lecture-translator.kit.edu>

<sup>2</sup><https://git.scc.kit.edu/isl/lt-middleware/ltpipeline>



- A comprehensive evaluation of different translation approaches and streaming algorithms

In the next section, we describe the overall architecture of the framework. The two following sections explain the streaming algorithms for the speech and text processing components. After that, we illustrate how we evaluate our framework and then how the experimental setup looks like. In Section 7 we present the results. Then, we review the related work. At the end we describe the limitations and conclude our work.

## 2 Dynamic Framework for low-latency speech translation

Motivated by previous work (Cho et al., 2013), we use a central mediator that coordinates the interaction of the different components (see Figure 1). The user sends data to an API component which then sends the data to the mediator. The mediator forwards all arriving data to the corresponding component(s), e.g., the audio signal from the user to the speech processing component, the resulting transcripts to the text processing component and the output (through the API) to the user. In order to allow a flexible processing, for each session a graph dynamically defines how the data is sent to the different components. We process different requests at each component using the existing streaming framework Kafka<sup>3</sup>.

Each component consists of a middleware and a backend with the processing separated into three steps:

1) **Input processing:** The middleware implements the streaming algorithms and can be run on the CPU. It uses the state of the current session to generate requests to the backend. Other approaches (Niehues et al., 2018) repeatedly send requests to the backend for all input messages. This can result in increasing latency if the backend is not able to keep up in high-load situations. In order to minimize this, we enable the middleware to skip intermediate processing steps. This is done by combining multiple input messages by concatenating audio or text. Several middleware workers can be run in parallel. We achieve the locality of the state by sticky queues, where a message from the same session is always sent to the same middleware worker.

2) **Backend request:** The backend contains the hosted models. It processes the requests without

<sup>3</sup><https://kafka.apache.org>

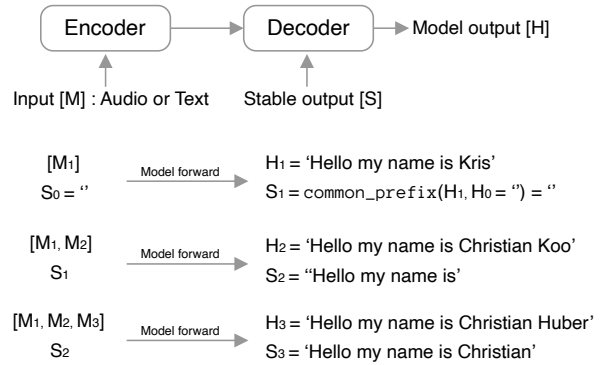


Figure 2: Stability detection

additional state information, is flexible to run on any device and is shared between different sessions. Because of the division in a stateful middleware and a stateless backend, we are able to share the backend and use batching of the requests.

3) **Output processing:** The output of a backend request is used to send information to the next component(s). Furthermore, the state of the corresponding session is updated.

Our framework supports two modes for low-latency speech translation. First, a revision mode (Niehues et al., 2018) where the component (Automatic speech recognition (ASR) or machine translation (MT)) can send stable and unstable outputs. Given more context at a later time step, the component can revise the unstable outputs. Second, a fixed mode (Liu et al., 2020a; Polák et al., 2022) where the component is only allowed to send stable output. For fixed mode (and the revision mode of the ASR component), the component needs to perform a stability detection (see Sections 3 and 4 and Figure 2), i.e., determine which parts of the output should be considered stable. Note that for our streaming algorithms the backend models need to support prefix decoding, i.e., one can send a prefix which is then forced in the output.

Our framework is easily extendable by deploying additional backend models for different languages, adding new streaming algorithms in the middleware or adding custom components (e.g., speaker diarization as a preprocessing step before the ASR) and including them in the session graph.

## 3 Low-latency Speech Processing

The speech processing component receives a stream of audio packets and sends chunks of text (transcript or translation) to the mediator. For this two steps are run:

**Input processing:** First, a voice activity detec-

tion generates a speech segment that can be extended when new packets of audio arrive. For this we use WebRTC Voice Activity Detector (Wiseman, 2016). Each audio frame (30ms) is classified if it contains speech or not. Then a moving average is calculated. If it exceeds a certain threshold, a new segment is started. New audio is added to this segment until the moving average falls below a certain threshold and the segment ends. Second, the backend model (ASR or speech translation (ST)) is run. If there exist speech segments that already ended, they are processed only once and the output is sent as stable text, other segments are constantly processed until they end.

**Stability detection and output processing:** We use the method local agreement two (LA2) from Polák et al. (2022). The intuition is that if the prefix of the output stays the same when adding more audio, the prefix should be considered stable. Let  $C$  denote the chunk size hyperparameter ( $LA2\_chunk\_size$ ). The fixed mode works as follows (see Figure 2): It waits until the segment contains (at least)  $C$  seconds of audio (denoted by  $M_1$ ) and then runs the model but does not output any stable text. Let's denote this first model output by  $H_1$ . After the segment contains (at least)  $C$  more seconds of audio (denoted by  $M_2$ ) the model is run again with all the audio and outputs  $H_2$ . Then the component outputs the common prefix of  $H_1$  and  $H_2$  as stable output  $S_2$ . After the segment again contains (at least)  $C$  more seconds of audio (denoted by  $M_3$ ) the model is run again with all the audio. However, now  $S_2$  is forced as prefix in the ASR/ST model decoding. The model outputs  $H_3$  and the common prefix from  $H_2$  and  $H_3$  is the next stable output  $S_3$ . This procedure is continued until the speech segment ends.

Note that the ASR/ST model has a certain maximum input size due to latency, memory and compute constraints. Therefore, if this limit is reached, the input audio to the model as well as the corresponding forced prefix is cut away.

The revision mode differs from the fixed mode in that the last hypothesis except the common prefix is sent as unstable output. Furthermore, in the time period until the speech segment contains again  $C$  more seconds of audio, the currently given audio is run through the model and the hypothesis except the last stable output is sent as unstable output.

## 4 Low-latency Text Processing

The text processing component receives a stream of (potentially revisable) text messages and sends chunks of text (translation) to the mediator.

**Input processing:** First, all input text that arrived is split into sentences by punctuation. Then, the backend model (MT) is run.

**Stability detection and output processing:** All sentences containing only stable text are processed once and the output is sent as stable text. For the other sentences containing unstable text the behavior depends on the mode. If text is stable or not is given by the speech processing component.

The revision mode works as follows: All sentences containing unstable text are processed by the backend model and the output text is sent as unstable text. A similar approach is not possible in the speech processing revision mode (see Section 3) since speech segments are not limited in size but the model input size is.

For the fixed mode we use the method local agreement from Liu et al. (2020a). The processing is similar to the speech processing. The difference is that the backend model is run when at least one new word is given instead of at least  $C$  seconds of audio. In our preliminary experiments, up to at least five words but the results were basically identical since the input is extended by a few words most of the time. Furthermore, only the stable part of the sentences containing unstable text is used as input. This restriction is not necessary in the speech processing component since there is no unstable audio input.

## 5 Evaluation Framework

We evaluate our system in an end-to-end fashion. That is, given an input audio, we send it to the system and evaluate the final returned transcript and translation. We provide an evaluation framework<sup>4</sup> that assess the system in different aspects and logs the results to categorized experiments on an UI board using MLflow (Zaharia et al., 2018). We consider different evaluation metrics as follows.

**BLEU:** In order to assess the translation quality, we use case-sensitive BLEU score, calculated using sacreBLEU (Post, 2018). We extract the final stable translation, align it sentence-wise with the gold reference using mwerSegmenter (Matusov et al., 2005) before calculating the BLEU score.

<sup>4</sup><https://git.scc.kit.edu/isl/lt-middleware/lt-evaluation>

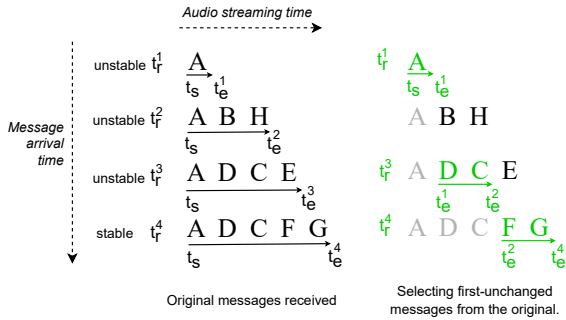


Figure 3: Example of collecting first-unchanged messages from an unstable-to-stable message block. The first-unchanged messages are in green.

**WER:** In order to assess the transcription quality of the ASR component in the cascaded setting, we use the case-sensitive Word Error Rate (WER) calculated using JiWER<sup>5</sup>. Similar as before, we extract the final stable transcription, align it sentence-wise with the gold reference using mwerSegmenter (Matusov et al., 2005) before calculating the WER.

**Latency:** We evaluate the latency of the system in an end-to-end manner. Factors such as network latency influence our latency metrics. However, our experiments are conducted locally, thus such factors are constant and negligible.

We define the end-to-end latency of the system as the average time (in seconds) it takes since an utterance is spoken until its first-unchanged translation is returned by the system. Note that the first-unchanged translation is not necessarily already marked as “stable” by the system.

For each message returned by the system, we have the stable/unstable flag along with three timestamps,  $t_s$ ,  $t_e$ ,  $t_r$ . The timestamps  $t_s$  and  $t_e$  are the start and end time of the audio segment that aligns to the message. The timestamp  $t_r$  is when the message was received. We collect the first unchanged messages as follows. We split the received messages into blocks of messages marked from “unstable” to “stable”. In each unstable-to-stable block, from the last stable message, we backtrack the previously received unstable messages to find the first ones that has prefix-overlaps with the final stable message. The illustration is shown in Figure 3.

Once we have collected the first-unchanged messages, we can calculate the latency. We use the same definition of delay as Niehues et al. (2016), where the average delay of the  $i^{th}$  message is:

$$d(t_s^i, t_e^i, t_r^i) = t_r^i - \frac{t_s^i + t_e^i}{2}.$$

<sup>5</sup><https://github.com/jitsi/jiwer>

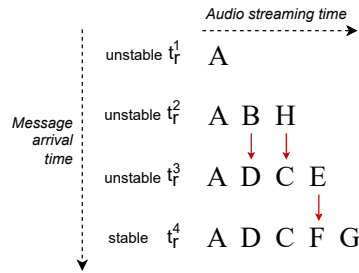


Figure 4: Example of flickers (denoted by red arrows) in an unstable-to-stable message block.

Then we calculate the latency as the weighted average of the delays of all  $m$  first-unchanged messages based on their length:

$$D = \frac{\sum_{i=1}^m d(t_s^i, t_e^i, t_r^i) * (t_e^i - t_s^i)}{\sum_{i=1}^m (t_e^i - t_s^i)}.$$

Note that the timestamps  $t_s$  and  $t_e$  in our latency formula are calculated by the used streaming algorithm. Therefore, we also tried another model-independent latency metric that only uses  $t_r$ . This metric approximates the segment-message alignment by assuming that each word output by the system has the duration of 0.3 second in the audio. Due to the strong assumption, this metric does not represent well the perceived latency. We only use this metric in order to verify our main model-dependent latency metric.

We find that the model-independent latency metric and our model-dependent metric provide the same relative ranking of the systems. This indicates that the timestamps  $t_s$  and  $t_e$  provided by the model itself are reliable to measure latency.

**Flickering rate:** The flickering rate is the average number of flickers per reference word. We count the number of flickers by looking at every pair of consecutive messages in a message block. If two words in the same position in the two messages differ, then it is counted as a flicker (see Figure 4). The flickering rate is calculated as the total number of flickers divided by the total number of words in the reference.

## 6 Experimental setup

### 6.1 Evaluation data

We test our system using datasets from different language pairs. Test datasets includes:

- Test data from the IWSLT shared task (*tst19*, *tst20*) (Anastasopoulos et al., 2021, 2022), where the domain is TED talks.

| Test data | Lang. pair | Hours | # Utt. |
|-----------|------------|-------|--------|
| tst19     | en→de      | 4.82  | 2279   |
| tst20     | en→de      | 4.09  | 1804   |
| LT CS     | de→en      | 6.39  | 2454   |
| LT nonCS  | de→en      | 2.66  | 1516   |
| mTEDx     | es→en      | 2.07  | 1012   |
|           | it→en      | 2.16  | 999    |
| ACL dev * | en→X       | 0.95  | 468    |

Table 1: Statistics of the test data. \*Test data containing en audio with translations into de, ja, zh, ar, nl, fr, fa, pt, ru and tr.

| Testset | C  | Cascaded ST |      |     | E2E ST |     |
|---------|----|-------------|------|-----|--------|-----|
|         |    | W ↓         | B ↑  | L ↓ | B ↑    | L ↓ |
| tst19   | .5 | 20.8        | 21.6 | 3.6 | 20.5   | 2.1 |
|         | 1  | 17.0        | 24.6 | 5.6 | 22.8   | 2.6 |
|         | 2  | 16.4        | 25.5 | 6.8 | 23.2   | 3.9 |
|         | 3  | 16.6        | 25.7 | 7.8 | 23.6   | 5.0 |
| ACL dev | .5 | 18.7        | 29.8 | 4.3 | 22.4   | 2.1 |
|         | 1  | 16.7        | 32.6 | 6.2 | 25.4   | 2.7 |
|         | 2  | 16.7        | 34.2 | 7.4 | 26.5   | 4.0 |
|         | 3  | 17.2        | 35.2 | 8.6 | 26.2   | 5.3 |

Table 2: Quality vs. latency (in fixed mode). C: *LA2\_chunk\_size* (s), W: WER, B: BLEU score, L: Latency (s) of the translation output.

- The test split of Multilingual TEDx Corpus (*mTEDx*) (Salesky et al., 2021), where the domain is TED talks.
- Lecture data (*LT*) which we collected internally at our university. This test set include a *CS* variance which includes lectures on the Computer Science domain, and a *nonCS* variance which includes lectures outside of the Computer Science domain.
- ACL development (*ACL dev*) set (Salesky et al., 2023), where the domain is ACL conference talks.

The detailed statistics of the test data is shown in Table 1.

## 6.2 Transcription and translation models

The English ASR models are built based on pretrained WavLM (Chen et al., 2022) and BART (Lewis et al., 2019)<sup>6</sup>, while for Multilingual ASR we utilized the XLS-R models (Babu et al., 2021) for the encoder and the MBART-50 model (Liu et al., 2020b) for the decoder following (Pham et al., 2022). On the other hand, the translation models are based on the pretrained

<sup>6</sup>With the recipe available at [here](#).

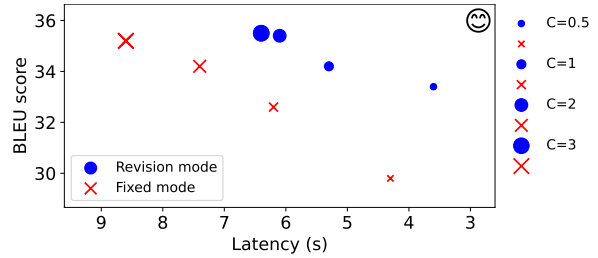


Figure 5: Latency vs. quality (for the cascaded model) in revision mode or fixed mode. C: *LA2\_chunk\_size* (s).

DeltaLM (Ma et al., 2021). For the en→X direction, the models are fine-tuned to optimize for ACL talks based on Liu et al. (2023). For other directions, DeltaLM is fine-tuned on the combination of commonly available datasets<sup>7</sup>.

Finally, for the end-to-end ST system, we used the language-agnostic model from Huber et al. (2022) that can decode en-de ST and de ASR.

## 7 Results and Discussion

### 7.1 Quality vs Latency trade-off

In the first experiment, we assess the trade-off between translation quality and latency by modifying the *LA2\_chunk\_size* parameter. The results are shown in Table 2. As can be seen, as we increase chunk size, the translation quality improves while the latency gets worse, both for cascaded ST and end-to-end ST. This is expected, since higher chunk size means longer input given to the model at each step, thus the output has better quality due to having more context, while the latency gets worse due to more waiting time for collecting the input.

### 7.2 Revision mode vs fixed mode

Second, we report the results of comparing the revision mode to the fixed mode with different *LA2\_chunk\_size* values when performing cascaded translation on the en-de ACL dev set. As can be seen in Figure 5, in general, revision mode has better BLEU score yet worse latency than fixed mode. This is expected, since for the revision mode, when more input audio is available, the system can correct its previous output, thus ending up having better translation quality yet worse latency due to the additional re-translation overhead.

<sup>7</sup>Paracrawl, UNPC, EUBookshop, MultiUN, EuroPat, TildeMODEL, DGT, Europarl, QED and NewsCommentary.

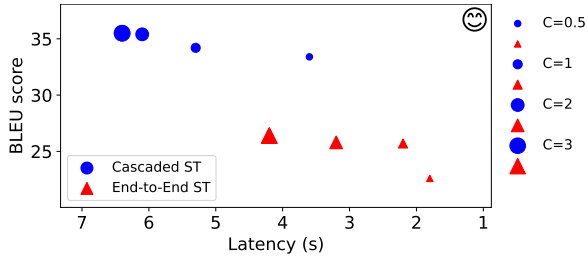


Figure 6: Latency vs. quality (in revision mode) for the cascaded ST or End-to-End ST model. C: *LA2\_chunk\_size* (s).

### 7.3 Cascaded vs End-to-End

Third, we report the results of comparing the cascaded setting to the end-to-end setting when performing online translation with revision mode on the ACL dev set. As can be seen in Figure 6, in general, cascaded ST has better BLEU score yet worse latency than end-to-end ST. Cascaded ST has worse latency since it contains two components and each component has to do computation. However, we observe that, with a similar latency of  $\sim 3.5$  seconds, cascaded ST still obtains a better BLEU score. On the other hand, end-to-end ST has a better minimum latency that can be achieved (almost two seconds lower than the cascaded system).

### 7.4 Load balancing

In order to assess the system’s capability to balance loads, we conduct experiments on running multiple sessions simultaneously using the same hosted model, with and without scaling the system’s number of middleware workers. For speech processing, we test parallel sessions on ACL dev en-de using the end-to-end ST model. For text processing, we test one cascaded ST session on ACL dev where the number of parallel sessions is the number of requested MT languages. In all experiments, we set *LA2\_chunk\_size* = 2. We report only the en-de results.

The results are shown in Table 3. As expected, the latency gets worse as the number of parallel sessions increases. Using multiple middleware workers counteracts that to some extent by making sure that the backend model is always busy and not waiting for the next request. Furthermore, we see that when the number of parallel sessions increases, the flickering rate decreases. This is because during higher load, fewer requests are sent to the backend and we observe less flickering. Here our automatic load balancing can be seen in action.

|   |   | Speech processing |                |                | Text processing |                |                |
|---|---|-------------------|----------------|----------------|-----------------|----------------|----------------|
| w | s | B $\uparrow$      | L $\downarrow$ | F $\downarrow$ | B $\uparrow$    | L $\downarrow$ | F $\downarrow$ |
| 1 | 1 | 25.9              | 3.2            | 0.5            | 34.9            | 6.7            | 0.5            |
|   | 2 | 26.1              | 20.2           | 0.5            | 34.6            | 8.4            | 0.4            |
|   | 5 | 21.3              | 28.2           | 0.2            | 34.8            | 28.1           | 0.2            |
| 5 | 1 | 26.2              | 3.2            | 0.6            | 35.1            | 6.1            | 0.5            |
|   | 2 | 26.5              | 4.6            | 0.5            | 33.6            | 8.0            | 0.5            |
|   | 5 | 25.3              | 16.7           | 0.3            | 34.5            | 15.9           | 0.2            |

Table 3: Quality, latency and flickering rate when scaling the number of sessions (with one hosted model per language). w: number of middleware workers, s: number of parallel sessions, B: Quality (BLEU score), L: Latency (s), F: Flickering rate. *LA2\_chunk\_size* is set to 2 seconds.

## 8 Related work

SimulEval (Ma et al., 2020) provides an evaluation framework for low-latency simultaneous speech translation with a decoupled client-server architecture allowing to plug-in translation models and stability detection policies. As the main difference we leave the audio segmentation up to the model whereas Ma et al. (2020) rely on a pre-segmentation of the audio, we factor in the computational latency in addition to the model latency and explore the scaling behavior in multi-session scenarios, both for a more realistic deployment scenario. Similar to this work Franceschini et al. (2020) implement a low-latency speech translation pipeline, however, their architecture does not scale well to multiple sessions and is not well suited for end-to-end evaluation.

## 9 Limitations and Conclusion

Since we run and evaluate the experiments in a realistic real-world scenario, it is difficult to exactly reproduce the results. The experiments are non-deterministic, e.g., because of network latencies. Furthermore, the results depend on the speed of the used hardware, especially the used hardware for the backend models. Additionally, we expect that each streaming algorithm implemented returns start and end timestamps. This may not be the case for all streaming algorithms one could want to compare.

In conclusion, this paper presented a framework for running and evaluating low-latency speech translation under realistic conditions. The research opens up new possibilities for advancing low-latency translation systems and serves as a resource for researchers seeking to improve the latency and quality of real-time speech translation applications by being able to properly evaluate different models

and streaming algorithms.

## Acknowledgements

The projects on which this paper is based were funded by the Federal Ministry of Education and Research (BMBF) of Germany under the numbers 01IS18040A (OML) and 01EF1803B (RELATER).

## References

- Sweta Agrawal, Antonios Anastasopoulos, Luisa Bentivogli, Ondřej Bojar, Claudia Borg, Marine Carpuat, Roldano Cattoni, Mauro Cettolo, Mingda Chen, William Chen, Khalid Choukri, Alexandra Chronopoulou, Anna Currey, Thierry Declerck, Qianqian Dong, Kevin Duh, Yannick Estève, Marcello Federico, Souhir Gahbiche, Barry Haddow, Benjamin Hsu, Phu Mon Htut, Hirofumi Inaguma, Dávid Javorský, John Judge, Yasumasa Kano, Tom Ko, Rishu Kumar, Pengwei Li, Xutai Ma, Prashant Mathur, Evgeny Matusov, Paul McNamee, John P. McCrae, Kenton Murray, Maria Nadejde, Satoshi Nakamura, Matteo Negri, Ha Nguyen, Jan Niehues, Xing Niu, Atul Kr. Ojha, John E. Ortega, Proyag Pal, Juan Pino, Lonneke van der Plas, Peter Polák, Elijah Rippeth, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Yun Tang, Brian Thompson, Kevin Tran, Marco Turchi, Alex Waibel, Mingxuan Wang, Shinji Watanabe, and Rodolfo Zevallos. 2023. **FINDINGS OF THE IWSLT 2023 EVALUATION CAMPAIGN**. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 1–61, Toronto, Canada (in-person and online). Association for Computational Linguistics.
- 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 Nadejde, 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.
- Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexander Waibel, Changhan Wang, and Matthew Wiesner. 2021. **FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN**. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 1–29, Bangkok, Thailand (online). Association for Computational Linguistics.
- Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhota, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, et al. 2021. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*.
- 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*, 16(6):1505–1518.
- Eunah Cho, Christian Fügen, Teresa Hermann, Kevin Kilgour, Mohammed Mediani, Christian Mohr, Jan Niehues, Kay Rottmann, Christian Saam, Sebastian Stüker, and Alex Waibel. 2013. **A real-world system for simultaneous translation of German lectures**. In *Proc. Interspeech 2013*, pages 3473–3477.
- Dario Franceschini, Chiara Canton, Ivan Simonini, Armin Schweinfurth, Adelheid Glott, Sebastian Stüker, Thai-Son Nguyen, Felix Schneider, Thanh-Le Ha, Alex Waibel, et al. 2020. Removing european language barriers with innovative machine translation technology. In *Proceedings of the 1st International Workshop on Language Technology Platforms*, pages 44–49.
- Christian Huber, Enes Yavuz Ugan, and Alexander Waibel. 2022. Code-switching without switching: Language agnostic end-to-end speech translation. *arXiv preprint arXiv:2210.01512*.
- 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*.
- Danni Liu, Thai Binh Nguyen, Sai Koneru, Enes Yavuz Ugan, Ngoc-Quan Pham, Tuan Nam Nguyen, Tu Anh Dinh, Carlos Mullov, Alexander Waibel, and Jan Niehues. 2023. **KIT’s multilingual speech translation system for IWSLT 2023**. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 113–122, Toronto, Canada (in-person and online). Association for Computational Linguistics.
- 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.
- Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, Alexandre Muzio, Saksham Singhal, Hany Hassan Awadalla, Xia Song, and Furu Wei. 2021. Deltalm: Encoder-decoder pre-training for language generation and translation by augmenting pretrained multilingual encoders. *arXiv preprint arXiv:2106.13736*.
- Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. **SIMULEVAL: An evaluation toolkit for simultaneous translation**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.
- Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. **Evaluating machine translation output with automatic sentence segmentation**. In *Proceedings of the Second International Workshop on Spoken Language Translation*, Pittsburgh, Pennsylvania, USA.
- Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alex Waibel. 2016. Dynamic transcription for low-latency speech translation. In *Interspeech*, pages 2513–2517.
- Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. **Low-Latency Neural Speech Translation**. In *Proc. Interspeech 2018*, pages 1293–1297.
- Ngoc-Quan Pham, Tuan Nam Nguyen, Thai-Binh Nguyen, Danni Liu, Carlos Mullov, Jan Niehues, and Alexander Waibel. 2022. **Effective combination of pretrained models - KIT@IWSLT2022**. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 190–197, Dublin, Ireland (in-person and online). Association for Computational Linguistics.
- Peter Polák, Ngoc-Quan Ngoc, Tuan-Nam Nguyen, Danni Liu, Carlos Mullov, Jan Niehues, Ondřej Bojar, and Alexander Waibel. 2022. Cuni-kit system for simultaneous speech translation task at iwslt 2022. *arXiv preprint arXiv:2204.06028*.
- 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.
- Elizabeth Salesky, Kareem Darwish, Mohamed Al-Badrashiny, Mona Diab, and Jan Niehues. 2023. Evaluating Multilingual Speech Translation Under Realistic Conditions with Resegmentation and Terminology. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*. Association for Computational Linguistics.
- Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W Oard, and Matt Post. 2021. The multilingual tedx corpus for speech recognition and translation. *arXiv preprint arXiv:2102.01757*.
- John Wiseman. 2016. py-webrtcvad. <https://github.com/wiseman/py-webrtcvad>. Accessed on: 03-08-2023.
- Matei A. Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. 2018. **Accelerating the machine learning lifecycle with mlflow**. *IEEE Data Eng. Bull.*, 41:39–45.

## A Detailed results

We report the overall performance of our system on different test data and language pairs with different settings at Table 4. In this experiment, we use the cascaded setting with `LA2_chunk_size = 2`. As can be seen, the BLEU scores drop around by one point when we move from offline to online setting (in fixed mode a little more), depending on the language directions.

## B Additional information

A video demonstrating the system can be found here: [Video link](#)

|                       |       | Offline         | Online: Revision mode    |                      |                              | Online: Fixed mode       |                      |
|-----------------------|-------|-----------------|--------------------------|----------------------|------------------------------|--------------------------|----------------------|
|                       |       | BLEU $\uparrow$ | $\Delta$ BLEU $\uparrow$ | Latency $\downarrow$ | Flickering rate $\downarrow$ | $\Delta$ BLEU $\uparrow$ | Latency $\downarrow$ |
| TED                   | tst19 | 27.2            | -1.3                     | 5.4                  | 0.5                          | -1.7                     | 5.3                  |
| (en $\rightarrow$ de) | tst20 | 29.8            | -1.1                     | 5.1                  | 0.5                          | -1.5                     | 6.9                  |
| LT                    | CS    | 25.2            | -2.0                     | 5.6                  | 0.6                          | -2.4                     | 6.0                  |
| (de $\rightarrow$ en) | nonCS | 28.5            | -0.2                     | 7.1                  | 0.6                          | -1.2                     | 5.7                  |
| mTEDx                 | es    | 31.0            | -2.5                     | 7.5                  | 0.4                          | -2.6                     | 7.6                  |
| (X $\rightarrow$ en)  | it    | 31.5            | -4.2                     | 12.5                 | 0.5                          | -5.1                     | 11.6                 |
| ACL dev               | de    | 36.5            | -1.9                     | 6.6                  | 0.5                          | -2.0                     | 7.5                  |
| (en $\rightarrow$ X)  | ja    | 39.6            | -1.6                     | 8.4                  | 0.1                          | -5.2                     | 8.7                  |
|                       | zh    | 45.3            | -0.5                     | 8.0                  | 0.1                          | -4.6                     | 8.0                  |
|                       | ar    | 28.3            | -0.8                     | 6.8                  | 0.5                          | -1.1                     | 7.3                  |
|                       | nl    | 42.7            | -1.3                     | 6.4                  | 0.5                          | -2.5                     | 7.4                  |
|                       | fr    | 43.7            | -0.4                     | 5.9                  | 0.5                          | -1.0                     | 7.6                  |
|                       | fa    | 21.8            | -0.8                     | 6.8                  | 0.7                          | -1.8                     | 7.5                  |
|                       | pt    | 45.2            | -1.2                     | 6.0                  | 0.4                          | -1.8                     | 7.3                  |
|                       | ru    | 13.5            | -1.0                     | 6.5                  | 0.5                          | -1.2                     | 7.4                  |
|                       | tr    | 20.1            | -0.9                     | 6.6                  | 0.5                          | -1.2                     | 7.4                  |

Table 4: Overall performance of our cascaded system with *LA2\_chunk\_size* set to 2 seconds: Quality, latency and flickering rate.  $\Delta$ BLEU: difference compared the corresponding offline setting.



# CHATREPORT: Democratizing Sustainability Disclosure Analysis through LLM-based Tools

Jingwei Ni<sup>1,2</sup> Julia Bingler<sup>3,4</sup> Chiara Colesanti-Senni<sup>1</sup> Mathias Kraus<sup>5</sup>  
Glen Gostlow<sup>1</sup> Tobias Schimanski<sup>1</sup> Dominik Stammach<sup>2</sup> Saeid Ashraf Vaghefi<sup>1,6</sup>  
Qian Wang<sup>1</sup> Nicolas Webersinke<sup>5</sup> Tobias Wekhof<sup>1,2</sup> Tingyu Yu<sup>1</sup> Markus Leippold<sup>1,7</sup>

<sup>1</sup>University of Zurich <sup>2</sup>ETH Zurich <sup>3</sup>University of Oxford

<sup>4</sup>Council on Economic Policies <sup>5</sup>FAU Erlangen-Nürnberg

<sup>6</sup>Eawag: Swiss Federal Institute of Aquatic Science <sup>7</sup>Swiss Finance Institute (SFI)

njingwei@ethz.ch

## Abstract

In the face of climate change, are companies really taking substantial steps toward more sustainable operations? A comprehensive answer lies in the dense, information-rich landscape of corporate sustainability reports. However, the sheer volume and complexity of these reports make human analysis very costly. Therefore, only a few entities worldwide have the resources to analyze these reports at scale, which leads to a lack of transparency in sustainability reporting. Empowering stakeholders with LLM-based automatic analysis tools can be a promising way to democratize sustainability report analysis. However, developing such tools is challenging due to (1) the hallucination of LLMs and (2) the inefficiency of bringing domain experts into the AI development loop. In this paper, we introduce CHATREPORT, a novel LLM-based system to automate the analysis of corporate sustainability reports, addressing existing challenges by (1) making the answers traceable to reduce the harm of hallucination and (2) actively involving domain experts in the development loop. We make our methodology, annotated datasets, and generated analyses of 1015 reports publicly available.<sup>12</sup>

## 1 Introduction

As climate change becomes an increasingly urgent issue, sustainability is becoming a key global concern, necessitating transparent public oversight of corporate sustainability practices. However, the substantial length of sustainability reports (often more than 70 pages) makes it challenging for the majority of stakeholders (including investors, policymakers, and the general public) to digest and analyze them. At the same time, relying on third-party rating agencies is not always a solution, as their services can be expensive, lack transparency,

and vary due to differing criteria for evaluating sustainability performance (Berg et al., 2022).

In light of these challenges, automated and transparent approaches are essential to improving accessibility, efficiency, and accuracy when analyzing corporate sustainability reports.

Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023a; OpenAI, 2023a; Touvron et al., 2023b, inter alia) have revolutionized Natural Language Processing (NLP), enabling advancements in automated reasoning, understanding, and generation of text. Such advances can assist in conducting comprehensive analyses of corporate sustainability reports automatically. However, to develop such an LLM-based system, there are two major challenges: LLMs (1) may hallucinate in their outputs (Ji et al., 2023), and (2) have no expertise in sustainability report analysis. Furthermore, there exists no framework which would actively involve domain experts in the prompt development loop, injecting domain expertise into the prompts.

In this paper, we propose CHATREPORT, a system that automatically analyzes sustainability reports based on the TCFD<sup>3</sup> (Task Force on Climate-related Financial Disclosures) recommendations. It computes the reports' conformity score to TCFD guidelines, proposing the first automatic metric for disclosure quality benchmarking. CHATREPORT also supports customized analysis with user question answering. To reduce hallucination, we ground the analytical prompts with retrieved information from the target report, and further make the answers traceable to help users identify hallucinations. To actively bring domain experts into the development loop, we design an automatic prompt engineering algorithm that transfers experts' feedback on specific outputs to general analysis guidelines, which

<sup>1</sup>Web app: <https://reports.chatclimate.ai/> Demo video: <https://www.youtube.com/watch?v=Q5AzaKzPE4M&t=15s>

<sup>2</sup><https://github.com/EdisonNi-hku/chatreport>

<sup>3</sup>We choose TCFD instead of other disclosure guidelines because it is widely adopted and investor-friendly. Appendix L covers an introduction for TCFD.

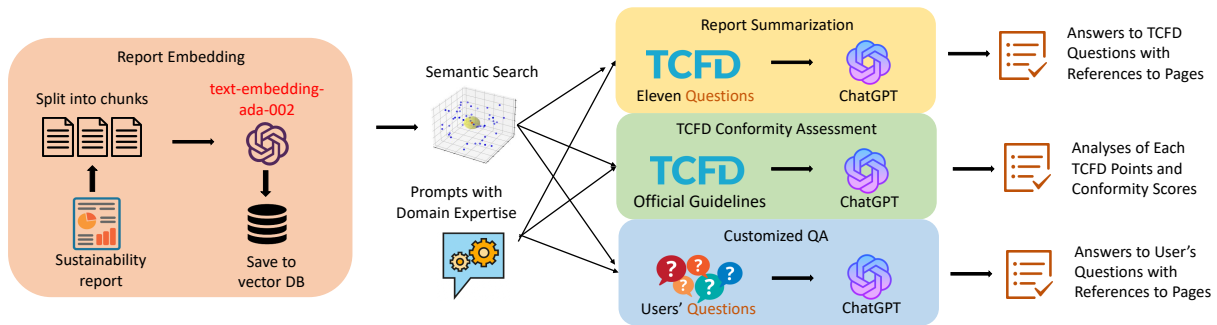


Figure 1: CHATREPORT Pipeline

can be injected into our prompt template for future analysis.

Furthermore, we conduct a rigorous human evaluation to analyze the system’s hallucination rate quantitatively. We find that the system achieves an admirable hallucination-free rate. For those hallucinated cases, it is easy for users to identify them because the system always (1) refers to relevant sources and pages; and (2) answers questions in an extractive manner, making it convenient to identify evidence sentences by keyword search. Moreover, we achieve a moderate inter-annotator agreement on annotating hallucination (Cohen’s Kappa of 0.54), further illustrating that the discrepancies between answers and references are easy to identify. Our human evaluation results in an annotated dataset of LLM outputs with attributions, which may contribute to other domains (e.g., LLM attribution verification (Yue et al., 2023) to check the supportiveness of cited sources for the answer.). Our contributions include:

1. We introduce CHATREPORT, a novel system that automatically analyzes sustainability reporting along different dimensions.
2. We develop an efficient framework to actively involve domain experts in AI tool development, which may potentially benefit all interdisciplinary research.
3. We conduct a human evaluation on CHATREPORT’s hallucination and attribution. The resulting dataset contributes to automatic attribution verification.

## 2 Related Work and Background

**NLP for Climate Change** NLP technologies have been employed in various areas, including meta-analyses in climate science (Callaghan et al., 2021), or for financial climate disclosure analyses (Bin-

gler et al., 2022a; Luccioni et al., 2020), detecting stance in media about global warming (Luo et al., 2020), detecting environmental claims (Stammbach et al., 2023), and climate claims fact-checking (Diggelmann et al., 2020; Webersinke et al., 2022). More recently, Vaghefi et al. (2023) introduced CHATCLIMATE, a chatbot based on the latest IPCC Assessment Report. By leveraging NLP, researchers aim to extract valuable insights from textual data related to climate change to advance research, decision-making, and public engagement.

**Large Language Models** LLMs have emerged as the de-facto standard in recent years (Brown et al., 2020; Ouyang et al., 2022; Chowdhery et al., 2022; Touvron et al., 2023a; Anil et al., 2023; OpenAI, 2023a; Touvron et al., 2023b). Instruction fine-tuned models, such as ChatGPT (OpenAI, 2023b) and GPT-4 (OpenAI, 2023a), have showcased their potential on comprehensive prompt-based AI applications (Shen et al., 2023; Schick et al., 2023). Some strong LLMs can even be a cheap and reliable proxy for human preference, evaluating the quality of generated texts (Chiang et al., 2023; Kocmi and Federmann, 2023; Zheng et al., 2023).

However, hallucination still remains a major limitation of the SOTA LLMs (Ji et al., 2023). Related work has proposed initial efforts to (1) better align LLMs (Zhou et al., 2023); and (2) fight false attribution from LLM-based search engine (Liu et al., 2023; Yue et al., 2023) and LLM-generated misinformation (Peng et al., 2023; Li et al., 2023). These efforts suggest potential ways to mitigate LLM hallucinations, but still left it as an open research question.

**Utilizing Experts’ Feedback** Involving a human in the loop has a long history in machine learning and NLP. However, previous work mainly focuses on active learning (Raghavan, 2006; Wu et al., 2021) and using human feedback to improve spe-

cific outputs (Elgohary et al., 2020; Tandon et al., 2021). In this work, we propose a novel prompting-based approach to automatically improve general prompts using experts’ feedback on specific outputs, which actively brings human experts into the prompt engineering loop.

### 3 CHATREPORT

#### 3.1 Pipeline

The pipeline of CHATREPORT is illustrated in Figure 1. Given a sustainability report, CHATREPORT analyzes it with the following four modules.

**Report Embedding (RE)** To address the limited context window, the RE module first splits the report into text chunks, which are then transformed into a vector space representation for future reference and semantic searching. We have domain experts transfer TCFD recommendations to queries for retrieval (details in Appendix E).

**Report Summarization (RS)** To assist in efficiently reading the report, the RS module summarizes it based on TCFD’s eleven recommended aspects that companies are asked to describe. Given each TCFD recommendation, the RS module first retrieves the relevant part from the report using our carefully designed query. Then it prompts the LLM to summarize the report’s disclosure on that TCFD recommendation, with the retrieved part (from the RE module) and the company’s basic information as context. Prompt templates for this module can be found in Appendix A.

**TCFD Conformity Assessment (TCA)** In addition to the recommendations, TCFD also provides detailed disclosure guidelines for each recommendation, which specify the type and granularity of information that companies need to disclose in their report. To evaluate the reports’ conformity to TCFD guidelines, we design the TCA module to analyze to which extent the report follows TCFD guidelines: for each TCFD recommendation, the TCA module takes in relative contexts from the RE module. It then evaluates it against the respective TCFD guidelines, generating an analysis paragraph and a TCFD conformity score from 0 to 100. The prompt template for this model can be found in Appendix D.

By explicitly defining the scoring criteria and providing clear instructions, we aim to minimize potential biases and enhance the reliability of the evaluation process. However, it is essential to ac-

knowledge that the LLM-generated scores might be far from perfect (Zheng et al., 2023). We believe that the scoring strategy implemented in our study represents a valid and valuable first step toward leveraging AI-based and automated methods for rating sustainability reports. We encourage future research and collaborative efforts to refine and improve this scoring strategy, considering alternative perspectives by including additional data sources and engaging a broader range of stakeholders.<sup>4</sup>

**Customized Question Answering (CQA)** Beyond the analytical structure provided by our framework, we enable users to conduct a personalized analysis by posing customized questions. Our prompt template takes in the user’s question and the retrieved relevant contexts which are queried by the question itself (using the RE module). Then, the CQA module makes an LLM call to answer the question. The CQA module’s prompt template is almost the same as the RS module’s question-answering prompt template, but with slightly different responding guidelines to deal with the noisier scenario where the questions are customized by the users (see details in Appendix C).

#### 3.2 Implementation Details

We use ChatGPT as the base LLM to conduct experiments and analysis in this paper. We use LangChain<sup>5</sup> to manage OpenAI API calls and vector-database retriever. We use OpenAI’s text-embedding-ada-002 for text chunk embedding. Empirically, we find that splitting reports into chunks of 500 characters (with an overlap of 20 characters between chunks) results in the best retrieval performance. We usually retrieve the top 20 related chunks from the RE module. If the prompt becomes too long (e.g., more than 4000 tokens) after inserting the retrieved chunks, we gradually remove the least relevant chunks until the prompt is suitable for the context window. We set the temperature to 0 for all LLM calls and reuse a static vector database for each report.

#### 3.3 Answer Traceability

To reduce hallucinations and improve interpretability, we attach source numbers to retrieved chunks

<sup>4</sup>We recall that our TCFD conformity score is not a rating or assessment of actual actions or commitments made by companies to address climate change. Instead, it measures the extent to which companies disclose relevant climate-related information in their financial reports.

<sup>5</sup><https://python.langchain.com/>

and prompt the LLM to provide its attribution (i.e., the chunks it refers to when summarizing information about TCFD recommendations and answering users’ questions). With the references attached, human experts can efficiently check whether the model produces misinformation. In Section 5.1, we quantitatively analyze the system’s answer traceability on a sampled set of outputs.

### 3.4 Expert-Involved Prompt Development

Prompt development is the critical part of CHATREPORT to make sure the outputs (1) contain granular details that stakeholders care about; (2) are formulated in an honest and traceable way; and (3) demonstrate awareness of potential cheap talk and greenwashing. To accomplish this, it is crucial for our domain experts to actively participate in prompt development. We first write several prompt templates, choosing one of them based on domain experts’ feedback on outputs. Then we empower domain experts with an LLM-based automatic prompt engineering tool, enabling them to fine-tune the prompts’ specifics autonomously, without the help of human prompt engineers. Details are described below:

**Prompt Template Selection: Question Answering or Summarization?** There are multiple ways to prompt an LLM to summarize a report’s disclosure regarding TCFD recommendations. One is to directly copy the original TCFD recommendation and prompt the LLM to summarize it (e.g., In governance, the company must describe the board’s oversight of climate-related risks and opportunities). Another is to first transfer the recommendation to a question about the report (see E for question-transformation details), then prompt the LLM to answer the question (e.g., How does the company’s board oversee climate-related risks and opportunities?). Our prompts for both scenarios are disclosed in Appendix A. We evaluate the prompt templates with experts involved, where the expert’s feedback shows that question answering outperforms disclosure summarization (one example is shown in Table 2).

**Automatic Prompt Engineering** Without granular adjustment on prompts, ChatGPT’s analysis of a sustainability report differs a lot from a human expert’s. For example, ChatGPT tends to flatter the user (due to its instruction-following nature), answering with optimism prior and becoming less critical of the possible cheap talk and greenwash-

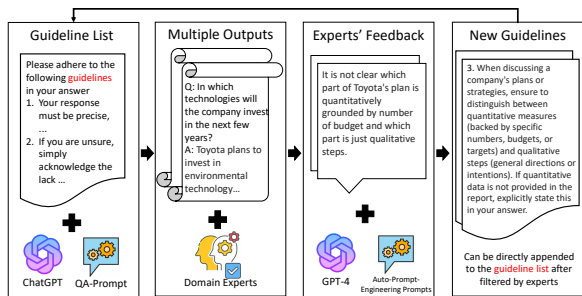


Figure 2: Automatic prompt engineering pipeline.

ing in the report. ChatGPT also tends to be wordy, including irrelevant or even hallucinated information in its response. Moreover, analysts usually expect critical information from a good summarization corresponding to TCFD recommendations, for example, the quantifiability and verifiability of the disclosure. However, ChatGPT fails to include such information because it is not explicitly stated in the TCFD recommendations. Analysts may even expect specific implicit information for each recommendation. To better incorporate these comprehensive, specific, and granular requirements in prompting, we design an automatic prompt engineering tool so that the domain experts can efficiently transfer their feedback on specific outputs to general analysis guidelines which can be used to improve the prompts.

The workflow of automatic prompt engineering is illustrated in Figure 2. The domain experts first suggest improvements for specific answers. Then we prompt ChatGPT to transform the feedback into guidelines that can be used to guide future TCFD question answering. In our prompt template, there is a list of guidelines that the LLM needs to adhere to in its answer. The generated guidelines are then appended to this list to improve the prompts. We started with a guideline list containing general guidelines for honest question answering:

Please adhere to the following guidelines in your answer

1. Your response must be precise, thorough, and grounded on specific extracts from the report to verify its authenticity.
2. If you are unsure, simply acknowledge the lack of knowledge, rather than fabricating an answer.

Then we develop new guidelines based on answers generated with this guideline list. Finally, we select five general guidelines for all question answering and one specific guideline for each TCFD recommendation. The general and specific guidelines can be found in Appendix C. Prompts of automatic prompt engineering can be found in Appendix B.

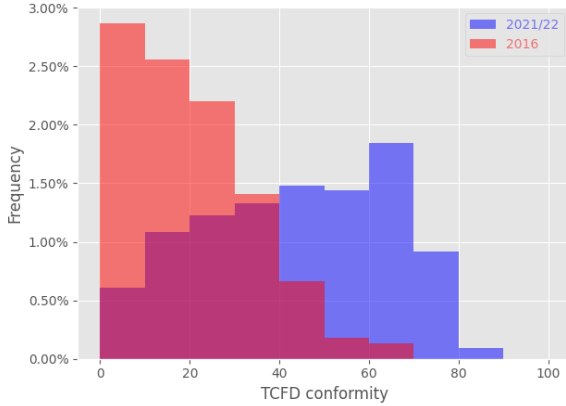


Figure 3: Density plot of the TCFD conformity for an arbitrary sample of corporate sustainability reports of companies listed on NYSE, sampled from 2016 and 2021/22.

### 3.5 Feedback Collection

We regard CHATREPORT as an ongoing learning system instead of a static analysis tool. Besides our domain experts, we also want to engage our users in the development and learning loop. We will collect users’ feedback on TCFD disclosure summarization and TCFD conformity analysis. Such feedback can either be used for both prompt improvements using our automatic prompt engineering method or be saved for memory and reflection for future refinements (Tandon et al., 2022).

## 4 Usages

We collected 9781 sustainability reports spanning 2010 to 2022 (fiscal years). Most of the reports are companies that are traded on the NASDAQ and NYSE. We find that the number of pages in corporate sustainability reports has slightly increased over recent years: in the fiscal year 2017, the mean length of the report has been at 59 pages. In 2021, this number increased to 70 pages, illustrating the increasing effort required by analyzing the reports manually.

**TCFD Conformity Analysis** Using the RS and TCA modules, we summarize TCFD disclosures and compute TCFD conformity scores for 1015 sustainability reports. Among these reports, 777 are from 2021 and 2022, while 227 are from 2016. Figure 3 illustrates the distribution of these scores for the two sample sets. Our findings indicate a significant impact of the TCFD recommendations on the average TCFD conformity, suggesting that companies embrace these guidelines. How-

| Backbone | Content      | Source       | R1/R2/RL                 |
|----------|--------------|--------------|--------------------------|
| ChatGPT  | <b>83.63</b> | <b>75.00</b> | 69.89/35.12/51.48        |
| GPT-4    | 69.09        | 72.37        | <b>85.20/50.31/61.50</b> |

Table 1: The **Content** column shows the hallucination-free rate on the content dimension. The **Source** column shows the hallucination-free rate on the source dimension when the answer is not hallucinated in content. **R1/R2/RL** shows the ROUGE-X precision scores using the retrieved report content as references. ChatGPT results are obtained on June 28th, 2023. GPT-4 results are obtained on July 6th, 2023.

ever, it is essential to note that TCFD conformity does not necessarily reflect the genuine commitment of companies toward their climate mitigation goals. It is necessary to consider the possibility of “cheap talk,” where firms may make superficial claims without substantial actions to address climate-related issues (Bingler et al., 2022b). In Appendix H, we showcase TCFD conformity analyses on sustainability reports of JP Morgan Chase, Shell and UBS in detail to illustrate the analytic usage of CHATREPORT.

**Customized Analysis** The CQA module allows users to customize their analysis through question-answering. Appendix G provides some illustrative examples of valuable analytic questions. Posing these questions allows us to gain valuable insights from the sustainability reports beyond the TCFD requirements summarized by the RS modules.

## 5 Hallucination Analysis

We conduct a human evaluation to assess the frequency and degree of hallucinations in CHATREPORT’s output when answering questions.<sup>6</sup> Hallucination is evaluated along two dimensions: (1) **Content**: An answer is not hallucinated if all its covered information is supported by the report. All answers that are not fully supported (e.g., extrapolation or partial support) are considered hallucinated on the content dimension. (2) **Source**: An answer is not hallucinated on the source dimension only when the model honestly reports its references and the content is not hallucinated; otherwise, the answer is hallucinated on the source dimension (we use binary annotation inspired by (Krishna et al., 2023)).

We randomly sampled 10 sustainability reports (110 TCFD question-answering pairs in total) for

<sup>6</sup>It is important to note that we analyze the answers’ honesty instead of quality.

human evaluation (sampling details in Appendix F). We have two different annotators to annotate each answer. If there is a disagreement on labeling, we assign a third annotator to make the decision. We conduct human evaluations on both ChatGPT and GPT-4 as the backbone LLM.

We surprisingly find that despite our strict annotation standard, CHATREPORT reaches a satisfactory hallucination-free rate. With ChatGPT, it honestly conveys information from the report **83.63%** of the time, considering the **51.5%** average hallucination-free rate of existing generative search engines reported by Liu et al. (2023).<sup>7</sup> Further findings and discussion are presented in the following subsections.

### 5.1 Answer Traceability

We find that CHATREPORT follows our instructions well by answering questions through copying or close paraphrasing. Table 1 shows that the answers achieve a high ROUGE precision score against the report content no matter with which backbone LLM, illustrating that the answers tend to adhere to the reports’ original utterances. This makes the outputs easy to trace using a simple keyword search. If a piece of information is not entailed by its evidence sentence, we mark it as hallucinated. An example can be found in Appendix J.

### 5.2 How Does CHATREPORT Hallucinate?

Most of the hallucinations on the content dimension lie in extrapolating reference chunks. Here is an example where the answer falsely concatenates two separate chunks:

```
Retrieved chunks in a prompt:
Content: ... Assurant may incur additional
        costs associated with tracking
Source: 174

Content: climate hazards. Own Operations: In
        addition to those noted in ...
Source: 186

... (more chunks and their source numbers)

LLM Answer:
... Risks include additional costs associated
        with tracking climate hazards, declining
        property values due to sea-level rise...
```

Although we explicitly told the model that the retrieved chunks might contain incomplete sentences at both ends and the chunks are delineated by source numbers and new lines, the LLM occasionally falsely concatenates two chunks or makes

<sup>7</sup>We quote Liu et al.’s (2023) result for reference. These numbers are not fully comparable because of the differences in task and data. See more in Appendix I.

erroneous extrapolations based on incomplete sentences. We leave the mitigation and automatic detection of such hallucinations to later versions of CHATREPORT and future work.

### 5.3 Which Backbone LLM is More Suitable?

We surprisingly find that ChatGPT outperforms GPT-4 by a large margin in answer honesty. This is because GPT-4 tends to summarize information at a higher level and make unnecessary inferences when answering the questions, which leads to more hallucination in detail. Since we label an answer hallucinated even if there is only a minor error, many GPT-4 answers are labeled as hallucinated. A comparison between GPT-4 and ChatGPT answers is showcased in Appendix K.

It is also harder to identify GPT-4’s hallucination than ChatGPT. When annotating hallucinations, we reach a Cohen’s Kappa of **0.54** for ChatGPT but only **0.21** for GPT-4. Sometimes, the highly abstractive and paraphrased nature of GPT-4 outputs makes it hard even for our expert annotators to identify hallucinations (though GPT-4 uses more utterances from the original reports as illustrated by the ROUGE precision scores). Therefore, we use ChatGPT as the backbone LLM for CHATREPORT

### 5.4 The Annotated Dataset

Our human evaluation for hallucination results in an expert-annotated dataset with labels of whether the answer is fully supported by the references. Human evaluation of how attributable an answer is to its reference is expensive and time-consuming, future work may study how to automatize this evaluation process (Yue et al., 2023) and benchmark the algorithm using our dataset.

## 6 Conclusion and Future Work

We propose CHATREPORT for automatic sustainability report analysis and demonstrating its potential applications and implications. Our prompt development loop and annotated datasets about hallucination could positively transfer to other NLP and interdisciplinary research. CHATREPORT is an open-sourced ongoing project. Our future work will focus on (1) enhancing the retrieval module to provide more accurate contexts for generation, (2) developing automatic attribution-checking tools to fight hallucination, and (3) migrating from OpenAI models to local LLM for more controllable output.

## Ethical Considerations

**Generate False Information:** Model hallucination is still a significant unresolved problem in NLP. CHATREPORT also generates hallucinations and requires some manual efforts to trace the answer. Moreover, due to the imperfect retrieval module, CHATREPORT may ignore some relevant information. To avoid causing misinformation and disinformation, on one hand, we disclaim on our website that CHATREPORT’s outputs can only be used as references, and cannot be cited as evidence or factual claims. On the other hand, we are experimenting with different approaches to make the outputs more accurate and will release better versions in the future.

**Bias towards Firm Perspective:** A limitation of our approach is the inherent bias towards the firm’s perspective in the extracted information from corporate sustainability reports. As CHATREPORT relies solely on the provided information as reported by the firm, it may struggle to provide unbiased and critical responses to certain questions. To mitigate this limitation, we will explore methods to incorporate external perspectives and independent sources of information in future work. This can be achieved by integrating data from third-party assessments, public opinion surveys, or expert evaluations. By incorporating a broader range of perspectives and data inputs, the analysis can provide a more comprehensive and balanced understanding of corporate sustainability performance.

**Changing Behavior of OpenAI Models:** OpenAI continues to update their model, leading to a changing performance of CHATREPORT’s backbone model (Chen et al., 2023). This may lead to less or more hallucination rates than we reported. In future work, we will substitute the OpenAI closed-source models with our own LLM checkpoints, making the system more controllable and reproducible.

**Human annotation:** All human annotators are co-authors of this paper, including climate and NLP researchers who have full knowledge about the context and utility of the collected data. We adhered strictly to ethical guidelines, respecting the dignity, rights, safety, and well-being of all participants. There are no data privacy issues or bias against certain demographics with regard to the annotated data.

**License of the Tool:** We use Apache License 2.0

to enable all stakeholders to use and adapt the Tool.

## Broader Implications

### Supporting Stakeholder Decision-making:

Stakeholders, including investors, customers, employees, and regulatory bodies, heavily rely on corporate sustainability reports to make informed decisions. The automated analysis provided by the framework empowers stakeholders with valuable insights into a company’s sustainability performance. Investors can use the extracted indicators to assess the environmental, social, and governance (ESG) risks and opportunities. Customers can make more sustainable choices by considering a company’s sustainability practices. Employees can evaluate a company’s commitment to social and environmental responsibility. Regulators can use the analysis results to monitor compliance with sustainability regulations.

### LLMs Disruptive Potential for the Rating Industry:

Empowering all stakeholders with an automated analysis framework could significantly diminish the need to rely on rating agencies for sustainability report assessments. This shift in power from rating agencies to the general public and investors can potentially disrupt rating agencies’ business models and challenge their long-standing dominance in sustainability reporting analysis: Rating agencies might start to focus on critical assessments of the information disclosed by companies, and provide external analyses of the strategies, rather than summarizing the firm’s information.

## Acknowledgements

This paper has received funding from the Swiss National Science Foundation (SNSF) under the project ‘How sustainable is sustainable finance? Impact evaluation and automated greenwashing detection’ (Grant Agreement No. 100018\_207800).

## References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan

- Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. [Palm 2 technical report](#).
- Florian Berg, Julian F Koelbel, and Roberto Rigobon. 2022. Aggregate confusion: The divergence of esg ratings. *Review of Finance*, 26(6):1315–1344.
- Julia Anna Bingler, Mathias Kraus, Markus Leippold, and Nicolas Webersinke. 2022a. Cheap talk and cherry-picking: What ClimateBert has to say on corporate climate risk disclosures. *Finance Research Letters*, page 102776.
- Julia Anna Bingler, Mathias Kraus, Markus Leippold, and Nicolas Webersinke. 2022b. Cheap talk in corporate climate commitments: The role of active institutional ownership, signaling, materiality, and sentiment. *Swiss Finance Institute Research Paper*.
- 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.
- Max Callaghan, Carl-Friedrich Schleussner, Shruti Nath, Quentin Lejeune, Thomas R Knutson, Markus Reichstein, Gerrit Hansen, Emily Theokritoff, Marina Andrijevic, Robert J Brecha, et al. 2021. Machine-learning-based evidence and attribution mapping of 100,000 climate impact studies. *Nature climate change*, 11(11):966–972.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2023. [How is chatgpt’s behavior changing over time?](#)
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%\* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 5 June 2023).
- 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*.
- Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bu- lian, Massimiliano Ciaramita, and Markus Leip- pold. 2020. Climate-fever: A dataset for verifica- tion of real-world climate claims. *arXiv preprint arXiv:2012.00614*.
- Ahmed Elgohary, Saghar Hosseini, and Ahmed Has- san Awadallah. 2020. [Speak to your parser: Interac- tive text-to-SQL with natural language feedback](#). In *Proceedings of the 58th Annual Meeting of the Asso- ciation for Computational Linguistics*, pages 2065– 2077, Online. Association for Computational Lin- guistics.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. [Survey of hallu- cination in natural language generation](#). *ACM Com- puting Surveys*, 55(12):1–38.
- Tom Kocmi and Christian Federmann. 2023. [Large Language Models Are State-of-the-Art Evaluators of Translation Quality](#). ArXiv:2302.14520 [cs].
- Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. [LongEval: Guidelines for human evaluation of faithfulness in long-form summarization](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Lin- guistics*, pages 1650–1669, Dubrovnik, Croatia. As- sociation for Computational Linguistics.
- Miaoran Li, Baolin Peng, and Zhu Zhang. 2023. [Self- Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models](#). ArXiv:2305.14623 [cs].
- Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. [Evaluating verifiability in generative search engines](#).
- Alexandra Luccioni, Emily Baylor, and Nicolas Duch- ene. 2020. [Analyzing sustainability reports using natural language processing](#).
- Yiwei Luo, Dallas Card, and Dan Jurafsky. 2020. [De- tecting stance in media on global warming](#). In *Find- ings of the Association for Computational Linguis- tics: EMNLP 2020*, pages 3296–3315, Online. As- sociation for Computational Linguistics.



- OpenAI. 2023a. [Gpt-4 technical report](#). Technical report, OpenAI.
- OpenAI. 2023b. [Instructgpt: Ai for generating instructions](https://openai.com/research/instructgpt/). <https://openai.com/research/instructgpt/>.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. [Check your facts and try again: Improving large language models with external knowledge and automated feedback](#).
- Hema Raghavan. 2006. Active learning with feedback on both features and instances.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-gpt: Solving ai tasks with chatgpt and its friends in huggingface](#).
- Dominik Stambach, Nicolas Webersinke, Julia Anna Bingler, Mathias Kraus, and Markus Leippold. 2023. Environmental claim detection. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, Canada*.
- Niket Tandon, Aman Madaan, Peter Clark, Keisuke Sakaguchi, and Yiming Yang. 2021. [Interscript: A dataset for interactive learning of scripts through error feedback](#).
- Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. [Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States. Association for Computational Linguistics.
- TCFD. 2017. Recommendations of the task force on climate-related financial disclosures. *Task force on climate-related financial disclosures*.
- TCFD. 2021. Implementing the recommendations of the tcf. *Task force on climate-related financial disclosures*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Christian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. [Llama 2: Open foundation and fine-tuned chat models](#).
- Saeid Ashraf Vaghefi, Qian Wang, Veruska Mucicione, Jingwei Ni, Mathias Kraus, Julia Bingler, Tobias Schimanski, Chiara Colesanti-Senni, Dominik Stambach, Nicolas Webersinke, et al. 2023. [Chatclimate: Grounding conversational ai in climate science](#).
- Nicolas Webersinke, Mathias Kraus, Julia Anna Bingler, and Markus Leippold. 2022. [Climatebert: A pretrained language model for climate-related text](#).
- Yuexin Wu, Yichong Xu, Aarti Singh, Yiming Yang, and Artur Dubrawski. 2021. [Active learning for graph neural networks via node feature propagation](#).
- Xiang Yue, Boshi Wang, Kai Zhang, Zirui Chen, Yu Su, and Huan Sun. 2023. [Automatic evaluation of attribution by large language models](#).
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-Bench and Chatbot Arena](#). ArXiv:2306.05685 [cs].
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srinu Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

## A Question Answering vs Summarization Prompts

Our two forms of prompt templates for the report summarization module are presented in this section. The content embraced by "{"}" are generic variables to be replaced by corresponding contents. For {basic\_info}. we provide the company's name, location, and sector, which are also retrieved by LLMs.

For {guidelines}, we use the guideline list developed by our experts (details in Appendix C). For {retrieved\_chunks\_with\_source}, we append each retrieved chunk with its chunk and page IDs for reference. For {A\_TCFD\_recommendation}, we use the original TCFD recommendations. For {question\_regarding\_a\_TCFD\_recommendation}, our experts rewrite each TCFD recommendation into a question form (details in Appendix E).

### A.1 Prompt for Question Answering

```
As a senior equity analyst with expertise in climate science evaluating a company's sustainability report, you are presented with the following background information:

{basic_info}

With the above information and the following extracted components (which may have incomplete sentences at the beginnings and the ends) of the sustainability report at hand, please respond to the posed question, ensuring to reference the relevant parts ("SOURCES").

Format your answer in JSON format with the two keys: ANSWER (this should contain your answer string without sources), and SOURCES (this should be a list of the source numbers that were referenced in your answer).

QUESTION: {
  question_regarding_a_TCFD_recommendation}
=====
{retrieved_chunks_with_source}
=====

Please adhere to the following guidelines in your answer:
{guidelines}

Your FINAL_ANSWER in JSON (ensure there's no format error):
```

### A.2 Prompt for Summarization

```
Your task is to analyze and summarize any disclosures related to the following <CRITICAL_ELEMENT> in a company's sustainability report:

<CRITICAL_ELEMENT>: {A_TCFD_recommendation}

Provided below is some basic information about the company under evaluation:

{basic_info}

In addition to the above, the following extracted sections (which may have incomplete sentences at the beginnings and the ends) of the sustainability report have been made available to you for review:

{retrieved_chunks_with_source}

Your task is to summarize the company's disclosure of the aforementioned <CRITICAL_ELEMENT>, based on the information presented in these extracts. Please adhere to the following guidelines in your summary:

{guidelines}

Your summarization should be formatted in JSON with two keys:
```

```
1. SUMMARY: This should contain your summary without source references.
2. SOURCES: This should be a list of the source numbers that were referenced in your summary.

Your FINAL_ANSWER in JSON (ensure there's no format error):
```

### A.3 Comparison between Two Prompts

The first row of Table 2 showcases an example of question-answering outperforms summarization according to experts' feedback. One explanation is that question answering explicitly tells the model what information is wanted while asking for disclosure summarization results in vague and superficial information.

## B Prompt for Automatic Prompt Engineering

The prompt for automatic prompt engineering takes in the prompt template, the old guideline list, AI's previous response, and an expert's feedback on the response. Then it comes up with a new guideline to enhance the current guideline list. The prompt is shown as follows:

```
You are a prompt engineer improving <Previous Prompt> given <Expert Feedback> and <AI's Previous Response>.

1. <Previous Prompt>: {"\"{original_prompt}

<Old Guideline List>: {guideline_list}
\"\"\"

2. <AI's Previous Response>: {"\"{old_response}\"\"

3. <Expert Feedback>: "{feedback}"

Given this feedback, could you please generate a new guideline that we can add to our existing list (<Old Guideline List>) to enhance future outputs? If <Expert Feedback> is already a guideline-like statement, keep its semantic while making it more generalize for future output.

Following are some examples of feedback-to-guideline transformation:

Expert Feedback1: <xxx information> is very important, please also analyze <xxx information> in the report.
Generated Guideline1: If the report provides <xxx information>, include it in the answer. Otherwise, explicitly state that the report does not cover <xxx information>.

Expert Feedback2: This answer includes some cheap talks in the report.
Generated Guideline2: If a piece of information looks like cheap talk, explicitly mark it as possible cheap talk in your answer.

The new guideline should be general enough for answering random question about random report. Avoid mention company-specific information in the guideline.
```

| Setting   | Example Outputs   | Expert's Comment   |
|---|---|--|
| Question Answering or Summarization?            | <p><b>Summarization:</b> Toyota's Board of Directors oversees climate-related risks and opportunities by monitoring progress towards targets, assessing risks multiple times a year, and formulating measures to solve climate-related issues. The committees are responsible for assessing and managing climate-related risks and opportunities. Reporting to the Board of Directors occurs every six months or when an important event arises.</p> <p><b>Question Answering:</b> The Board of Directors oversees climate-related risks and opportunities by deliberating and supervising strategies, major action plans, and business plans, monitoring progress toward qualitative and quantitative targets addressing climate issues, and considering the financial impact of risks and opportunities. The Environmental Product Design Assessment Committee and Production Environment Committee are the major bodies in assessing and managing climate-related risks and opportunities, reporting to the Board of Directors every six months or when an important event arises.</p>   | Both outputs include information on the board's oversights on risks and opportunities, the responsible committees, and the reporting mechanism. However, question-answering provides more detailed and granular information than summarization.                                |
| Answer Length                                   | <p><b>No length limitation:</b> Toyota uses multiple metrics to manage climate-related risks and opportunities, including CO2 emissions and other elements related to climate change. The metrics are integrated into the Toyota Environmental Challenge 2050, 2030 Milestone, and Seventh Toyota Environmental Action Plan. Toyota aims to achieve carbon neutrality by 2050 with three zero challenges. The company promotes electrification and renewable energy in production processes and takes adaptation measures for natural disasters. The risks relating to climate change are identified and assessed based on the TGRS, and important risks and opportunities are reported to the Board of Directors. However, the report lacks specific information on the targets and the company's performance against them.</p> <p><b>50 words:</b> Toyota uses multiple metrics to manage climate-related risks and opportunities, including CO2 emissions, energy, water, resource recycling, and biodiversity. The metrics are integrated into the Toyota Environmental Challenge 2050, 2030 Milestone, and Seventh Toyota Environmental Action Plan. Toyota aims to achieve carbon neutrality by 2050 with three zero challenges. The report does not provide sufficient information to evaluate the organization's performance against these targets.</p> | Both answers cover the key points about Toyota's metrics, targets, and lacked information. Longer responses tend to over-include details that are not directly asked for and are not necessarily more informative, which may exaggerate the richness of disclosed information. |
| With or without guidelines on critical thinking | <p><b>Without:</b> Toyota has identified risks and opportunities associated with climate change that may impact its businesses, strategy, and financial planning. The risks are classified into physical and transition risks, including tightening regulations for fuel efficiency and zero-emission vehicles, expansion of carbon pricing, and increase in frequency and severity of natural disasters. Toyota aims to achieve carbon neutrality by 2050 and has set multiple metrics to manage climate-related risks and opportunities.</p> <p><b>With:</b> Toyota has identified risks and opportunities associated with climate change, including physical risks such as storms and floods, and transition risks such as tightening regulations for fuel efficiency and carbon pricing. The company aims to achieve carbon neutrality by 2050 and has metrics in place to assess climate-related risks and opportunities. However, the report lacks concrete data on the impact of these risks and opportunities on the company's financial performance.</p>   | Both outputs summarize details about Toyota's identification of physical and transition risks and the company's goal. However, with critical thinking guidance, the output becomes more concise and raises a lack-of-detail drawback.  |

Table 2: Illustrative examples for the expert-involved development loop. Each row shows a prior output in the loop and an improved version after taking experts' advice in prompts. Corresponding information aspects are highlighted with the same color for clarity and comparison. Three rows of outputs correspond to the 1st (the company's board's oversight), 9th (metrics for assessing risks and opportunities), and 3rd (climate-related risks and opportunities) TCFD recommendations. We randomly pick Toyota's 2022 sustainable report for illustration. Similar phenomena can also be observed in other sustainability reports.

The new guideline should be concise and easy to follow by an AI assistant. Please format your answer in JSON with a single key "GUIDELINE"

Your answer in JSON (make sure there's no format error):

## C Guidelines for Question Answering

Using automatic prompt engineering, we come up with granular guidelines for question-answering prompts using experts' feedback, including five guidelines for all question-answering:

- Keep your ANSWER within {answer\_length} words.
- Be skeptical to the information disclosed in the report as there might be greenwashing (exaggerating the firm's environmental responsibility). Always answer in a critical tone.
- cheap talks are statements that are costless to make and may not necessarily

- reflect the true intentions or future actions of the company. Be critical for all cheap talks you discovered in the report.
- Always acknowledge that the information provided is representing the company's view based on its report.
  - Scrutinize whether the report is grounded in quantifiable, concrete data or vague, unverifiable statements, and communicate your findings.

And specific guidelines for different TCFD questions:

```
tcfd_guidelines = {
  'tcfd_1': "8. Please concentrate on the board's direct responsibilities and actions pertaining to climate issues, without discussing the company-wide risk management system or other topics.",
  'tcfd_2': "8. Please focus on their direct duties related to climate issues, without introducing other topics such as the broader corporate risk management system."
}
```

```

'tcfd_3': "8. Avoid discussing the company
-wide risk management system or how
these risks and opportunities are
identified and managed.",
'tcfd_4': "8. Please do not include the
process of risk identification,
assessment or management in your
answer.",
'tcfd_5': "8. In your response, focus
solely on the resilience of strategy
in these scenarios, and refrain from
discussing processes of risk
identification, assessment, or
management strategies.",
'tcfd_6': "8. Restrict your answer to the
identification and assessment
processes, without discussing the
management or integration of these
risks.",
'tcfd_7': "8. Please focus on the concrete
actions and strategies implemented to
manage these risks, excluding the
process of risk identification or
assessment.",
'tcfd_8': "8. Please focus on the
integration aspect and avoid
discussing the process of risk
identification, assessment, or the
specific management actions taken.",
'tcfd_9': "8. Do not include information
regarding the organization's general
risk identification and assessment
methods or their broader corporate
strategy and initiatives.",
'tcfd_10': "8. Confirm whether the
organisation discloses its Scope 1,
Scope 2, and, if appropriate, Scope 3
greenhouse gas (GHG) emissions. If so,
provide any available data or
specific figures on these emissions.
Additionally, identify the related
risks. The risks should be specific to
the GHG emissions rather than general
climate-related risks.",
'tcfd_11': "8. Please detail the precise
targets and avoid discussing the
company's general risk identification
and assessment methods or their
commitment to disclosure through the
TCFD.",
}

```

All these guidelines contribute to the answer quality. For example, the second and third row of Table 2 illustrate that restricting the answer length and adding guidelines for critical thinking improve the answering quality.

## D Prompt for TCFD Conformity Assessment

In the prompt employed for scoring company disclosures, we provide the following statement to guide the process of rating the TCFD conformity of the sustainability reports:

```

Your task is to rate a sustainability report's
disclosure quality on the following <
CRITICAL_ELEMENT>:

<CRITICAL_ELEMENT>: {tcfd_recommendation}

These are the <REQUIREMENTS> that outline the
necessary components for high-quality
disclosure pertaining to the <
CRITICAL_ELEMENT>:

<REQUIREMENTS>:
---
```

```

{requirements}
---

Presented below are select excerpts from the
sustainability report, which pertain to
the <CRITICAL_ELEMENT>:

<DISCLOSURE>:
---
{disclosure}
---
```

Please analyze the extent to which the given < DISCLOSURE> satisfies the aforementioned < REQUIREMENTS>. Your ANALYSIS should specify which <REQUIREMENTS> have been met and which ones have not been satisfied. Your response should be formatted in JSON with two keys:

1. ANALYSIS: A paragraph of analysis (be in a string format). No longer than 150 words.
2. SCORE: An integer score from 0 to 100. A score of 0 indicates that most of the < REQUIREMENTS> have not been met or are insufficiently detailed. In contrast, a score of 100 suggests that the majority of the <REQUIREMENTS> have been met and are accompanied by specific details.

Your FINAL\_ANSWER in JSON (ensure there's no format error):

Where "{requirements}" denote the TCFD official guidelines for disclosure; "{disclosure}" denotes the extracted relevant chunks from the report; and "{tcfd\_recommendation}" denotes the TCFD recommendation to be analyzed.

This prompt enables evaluators to systematically assess the disclosure quality of sustainability reports by assigning scores that reflect the level of detail and comprehensiveness in the disclosed information. While the scoring strategy employed is designed to assess the reports' TCFD conformity systematically, it is essential to acknowledge that no scoring approach can be perfect. Acknowledging the potential limitations and imperfections, we firmly believe that the scoring strategy implemented in our study represents a valid and valuable first step toward leveraging AI-based and automated methods for rating sustainability reports. Moreover, by explicitly defining the scoring criteria and providing clear instructions, we aim to minimize potential biases and enhance the reliability of the evaluation process. Nevertheless, we encourage future research and collaborative efforts to refine and improve this scoring strategy, considering alternative perspectives and engaging a broader range of stakeholders.

## E The Eleven TCFD Questions

Our domain experts rewrite the eleven TCFD recommendations (TCFD, 2017, 2021) into the following eleven questions:

GOVERNANCE

1. How does the company's board oversee climate-related risks and opportunities?
2. What is the role of management in assessing and managing climate-related risks and opportunities?

#### STRATEGY

3. What are the most relevant climate-related risks and opportunities that the organization has identified over the short, medium, and long term? Are risks clearly associated with a horizon?
4. How do climate-related risks and opportunities impact the organization's business strategy, economic and financial performance, and financial planning?
5. How resilient is the organization's strategy when considering different climate-related scenarios, including a 2°C target or lower scenario? How resilient is the organization's strategy when considering climate physical risks?

#### RISK MANAGEMENT

6. What processes does the organization use to identify and assess climate-related risks?
7. How does the organization manage climate-related risks?
8. How are the processes for identifying, assessing, and managing climate-related risks integrated into the organization's overall risk management?

#### METRICS AND TARGETS

9. What metrics does the organization use to assess climate-related risks and opportunities? How do these metrics help ensure that performance aligns with its strategy and risk management process?
10. Does the organization disclose its Scope 1, Scope 2, and, if appropriate, Scope 3 greenhouse gas (GHG) emissions? What are the related risks, and do they differ depending on the scope?
11. What targets does the organization use to understand, quantify, and benchmark climate-related risks and opportunities? How is the organization performing against these targets?

These questions are designed to extract specific information related to oversight, management, risks, opportunities, resilience, processes, metrics, disclosure, and targets concerning climate-related aspects within the organization.

## F Report Sampled for Hallucination Analysis

Using a random seed of 43, we sampled 10 reports for hallucination analysis: NYSE\_WMT\_2022.pdf, NYSE\_SE\_2021.pdf, NYSE\_PNC\_2021.pdf, NYSE\_PLD\_2016.pdf, NYSE\_PBR\_2016.pdf, NYSE\_ITT\_2019.pdf, NYSE\_FTV\_2022.pdf, NYSE\_JPM\_2021.pdf, NYSE\_BV\_2022.pdf, and NYSE\_AIZ\_2022.pdf. All these reports are available in our GitHub.

## G Customized Analysis Examples

This section provides some illustrative examples of possible questions that can be answered by CHARTREPORT based on the information available in the sustainability report of Sony and Shell, respectively. The specific questions and the detailed answers are provided in Table 4 and Table 5. Posing these questions allows us to gain valuable insights from the sustainability reports beyond the TCFD requirements.

With the questions posed for Sony (Table 4), we find that with respect to the compatibility of Sony's transition plan with a 1.5 degrees pathway, the report lacks sufficient information to make a determination. While the report mentions scenario analysis using 2°C and 4°C scenarios and Sony's commitment to reducing greenhouse gas emissions, it does not provide concrete data or targets to assess the plan's compatibility with a 1.5 degrees pathway. While it mentions the establishment of annual plans by individual business units and sites, incorporating guiding principles and regular progress reviews, it lacks specific details. Assessing the ambition of Sony's transition plan is challenging due to the report's lack of specific targets and progress data related to its environmental goals. However, Sony supports the TCFD and actively participates in the TCFD Consortium in Japan.

With the questions posed for Shell (Table 5), we observe that the report contains qualitative information. Still, its adequacy in presenting a comprehensive view of the company's sustainability efforts is uncertain. Moreover, we can identify instances of cheap talk, where unrelated topics like the war in Ukraine and the cost of living were mentioned, potentially diverting attention from the company's sustainability performance. Regarding the company's transition plan, the report lacks clarity on the specific interim steps taken, only mentioning responsible investments in the energy transition and screening investments against multiple criteria. Inconsistencies were noted in the review of global targets and the potential retirement of certain business KPI targets without clear implications provided. Additionally, the report acknowledges that Shell's operating plans cannot fully reflect their net-zero emissions and NCI targets, raising concerns about the feasibility of their transition plan.

| FY   | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Average | # pages |
|------|----|----|----|----|----|----|----|----|----|-----|-----|---------|---------|
| 2014 | 0  | 0  | 20 | 40 | 40 | 30 | 40 | 20 | 20 | 40  | 70  | 29.09   | 436     |
| 2015 | 0  | 20 | 20 | 30 | 50 | 20 | 40 | 20 | 20 | 30  | 40  | 26.36   | 529     |
| 2016 | 50 | 60 | 40 | 20 | 30 | 30 | 40 | 50 | 30 | 40  | 60  | 40.90   | 510     |
| 2017 | 10 | 30 | 20 | 10 | 40 | 20 | 40 | 40 | 0  | 60  | 60  | 30.00   | 561     |
| 2018 | 0  | 10 | 20 | 30 | 20 | 40 | 40 | 20 | 20 | 40  | 40  | 25.45   | 317     |
| 2019 | 60 | 60 | 40 | 40 | 40 | 70 | 60 | 70 | 30 | 50  | 50  | 51.81   | 214     |
| 2020 | 60 | 60 | 40 | 80 | 70 | 50 | 10 | 50 | 20 | 60  | 50  | 58.18   | 170     |
| 2021 | 60 | 60 | 40 | 70 | 70 | 60 | 70 | 80 | 30 | 60  | 60  | 60.00   | 199     |
| 2022 | 70 | 60 | 40 | 60 | 60 | 60 | 70 | 60 | 30 | 60  | 50  | 56.36   | 164     |

Table 3: SONY Scores on TCFD-conformity

## H TCFD Conformity Analysis Examples

The output of the TCFD analysis is structured according to the eleven questions from the TCFD report. This section provides the analysis summary of the corporate sustainability reports of JP Morgan Chase in 2015 and 2021, Shell in 2022, and UBS in 2022. For each of the eleven questions, we report the question itself, the answer to the question, and the analysis of the disclosed content, which gives the score. The total score of the report is calculated as the average over all eleven scores.

For JP Morgan Chase, CHATREPORT gives an average score of zero for the year 2015 (Appendix H.1, which is reassuring since the TCFD recommendations were not yet published and there is almost no environmental information in JP Morgan’s report. However, for the year 2021 (Appendix H.2), the score for JP Morgan reached more than 61, clearly indicating the adoption of TCFD guidelines in their climate-related disclosure. In Appendix H.3, we observe that while Shell acknowledges the significance of climate-related risks and opportunities, CHATREPORT argues that question five of the TCFD guidelines is not fully addressed by the report. CHATREPORT criticizes the lack of specific details on how the organization’s strategy would adapt to the potential impact of climate-related issues on financial performance under different climate-related scenarios. Moreover, the report provides no information on the resilience of Shell’s strategy when facing physical climate risks. Overall, the report’s disclosure on this TCFD recommendation point is deemed insufficient and therefore obtains a low score.

Inspecting UBS in Appendix H.4 as another example, we find that for question four of the TCFD report, the sustainability report provides some high-quality information (score of 90). In particular, the report provides a comprehensive overview of the impact of climate-related issues across various as-

pects, such as products and services, supply chain, adaptation and mitigation activities, and access to capital, among others. The report demonstrates a holistic understanding of the interconnectedness of these factors and their influence on the organization’s long-term value creation. Nevertheless, CHATREPORT argues that the report falls short in providing specific details on greenhouse gas emissions targets and concrete actions to reduce emissions in their operations and value chain or support the transition to a low-carbon future.

### H.1 JP Morgan Chase 2015

#### Governance:

Q1: How does the company’s board oversee climate-related risks and opportunities?

A1: The report does not provide specific information on how the company’s board oversees climate-related risks and opportunities. It only mentions that the board is responsible for governance issues within the business.

Analysis1: The sustainability report does not provide any information on how the company’s board oversees climate-related risks and opportunities. The report only contains vague and irrelevant information that does not meet any of the requirements for high-quality disclosure pertaining to the critical element. The report does not discuss the processes and frequency by which the board and/or board committees are informed about climate-related issues, whether the board and/or board committees consider climate-related issues when reviewing and guiding strategy, major plans of action, risk management policies, annual budgets, and business plans, or how the board monitors and oversees progress against goals and targets for addressing climate-related issues. As a result, the report receives a score of 0.

**Score1: 0**

---

**Question:** "Is the transition plan compatible with a 1.5 degrees pathway, i.e., Paris-aligned pathway?"

**Answer:** The report does not provide sufficient information to determine if Sony's transition plan is compatible with a 1.5 degrees pathway. The report only mentions scenario analysis using 2°C and 4°C scenarios, and Sony's commitment to reducing greenhouse gas emissions. However, there is no concrete data or targets provided to assess the compatibility of Sony's transition plan with a 1.5 degrees pathway.

**Pages:** [112, 98, 108]

---

**Question:** "What are the policies and activities that the company plans to implement in the next three years, i.e., short term, to implement its transition plan?"

**Answer:** Sony's sustainability report does not provide clear and specific policies and activities that the company plans to implement in the next 3 years to implement its transition plan. The report only mentions that individual business units and sites establish and implement their own annual plans, incorporating essential elements of guiding principles established by the headquarters, and progress on the implementation of these business plans is reviewed regularly by a committee. However, the report lacks concrete details on the specific policies and activities that will be implemented in the short term.

**Pages:** [107]

---

**Question:** "Does the company's transition plan lack ambition? What do you think?"

**Answer:** The report does not provide sufficient information to determine whether Sony's transition plan lacks ambition. While the report mentions Sony's Road to Zero environmental plan, it does not provide specific details on the plan's targets or how it will achieve them. Additionally, the report lacks concrete data on Sony's progress towards its environmental goals, making it difficult to assess the ambition of its transition plan.

**Pages:** [113, 110]

---

**Question:** "Does the company support the TCFD?"

**Answer:** Yes, Sony Corporation supports the TCFD and participates in the TCFD Consortium in Japan to facilitate implementation and discussion among companies and financial institutions that endorse the TCFD Recommendations. Sony also disclosed climate-related information in accordance with the TCFD Recommendations.

**Pages:** [113, 165, 166, 111]

---

Table 4: User-specific questions to a corporate sustainability report: Sony FY2020

Q2: What is the role of management in assessing and managing climate-related risks and opportunities?

A2: The report does not provide a clear answer on the role of management in assessing and managing climate-related risks and opportunities. There are no specific mentions of climate-related risks and opportunities in the report, and the content provided is not relevant to the question.

Analysis2: The given disclosure does not provide any information on the role of management in assessing and managing climate-related risks and opportunities. The report only contains general information about the organization's governance structure and its initiatives related to economic growth and social issues. The report does not mention any

---

**Question:** "What is the level of cheap talk in the report?"

**Answer:** The report contains some qualitative information, but it is not clear whether it is sufficient to provide a complete picture of the company's sustainability efforts. The report also includes some cheap talk, such as mentioning topical issues like the war in Ukraine and the cost of living, which may not be directly related to the company's sustainability performance.

**Pages:** [11]

---

**Question:** "What are the interim steps that the company takes to implement its transition plan?"

**Answer:** Shell's sustainability report does not provide clear information on the interim steps taken to implement its transition plan. The report only mentions that the company invests responsibly in the energy transition and screens its investments against multiple criteria. However, it is unclear what specific actions the company takes to transition to net-zero emissions.

**Pages:** [63]

---

**Question:** "What are possible inconsistencies in the firm's transition plan?"

**Answer:** The report mentions that existing global targets are under review, and some business KPI targets may be retired as part of normal strategy evolution. However, it is unclear what specific targets are being reviewed and what the implications of retiring them would be. Additionally, the report states that Shell's operating plans cannot reflect their 2050 net-zero emissions target and 2035 NCI target, which raises questions about the feasibility of their transition plan.

**Pages:** [88]

---

**Question:** "In which technologies will the company invest in the next few years?"

**Answer:** Shell plans to invest in low-carbon energy solutions such as biofuels, hydrogen, charging for electric vehicles, wind, and solar power. They invested \$8.2 billion in low-carbon energy and non-energy products, around a third of their total cash capital expenditure of \$25 billion in 2022. They also won bids with their partners to build four offshore wind farms in the Netherlands, the UK, and the USA, and extended their onshore wind activities into Australia and the Philippines. They acquired Indian solar developer Spring Energy and completed the acquisition of Nature Energy of Denmark, which is Europe's biggest producer of renewable natural gas made from agricultural, industrial, and household waste.

**Pages:** [24, 33, 3, 25]

---

Table 5: User-specific questions to a corporate sustainability report: Shell FY2022

specific management-level positions or committees responsible for climate-related issues, nor does it describe any processes by which management is informed about climate-related issues. Therefore, the report fails to meet any of the requirements for high-quality disclosure pertaining to the critical element of management's role in assessing and managing climate-related risks and opportunities.

**Score2: 0**

**Strategy:**

Q3: What are the most relevant climate-related risks and opportunities that the organisation has identified over the short, medium, and long term?

Are risks clearly associated with a horizon?

A3: The report does not provide a clear identification of climate-related risks and opportunities over the short, medium, and long term. The report focuses on the company's economic opportunity, job creation, and public-private partnerships. The report lacks concrete data and quantifiable information on the company's climate-related risks and opportunities.

Analysis3: The given disclosure does not provide any information on the most relevant climate-related risks and opportunities that the organization has identified over the short, medium, and long term. The disclosure does not meet any of the requirements outlined for this critical element. Instead, the report focuses on governance issues, economic opportunity, and lessons learned broadly. The report does not provide any information on the process used to determine which risks and opportunities could have a material financial impact on the organization. The lack of information on climate-related risks and opportunities makes it difficult to assess the organization's preparedness for climate change. Therefore, the report's disclosure quality is poor.

**Score3: 0**

Q4: How do climate-related risks and opportunities impact the organisation's businesses strategy, economic and financial performance, and financial planning?

A4: The sustainability report does not provide a clear and specific answer to the question. While the report highlights the company's commitment to economic growth and opportunity, it lacks concrete information on how climate-related risks and opportunities impact the organization's business strategy, economic and financial performance, and financial planning.

Analysis4: The given disclosure does not satisfy the requirements for the critical element. The report does not provide any information on how climate-related risks and opportunities impact the organization's business strategy, economic and financial performance, and financial planning. The report does not discuss the impact of climate-related issues on their financial performance and financial position. The report does not describe how climate-related issues serve as an input to their financial planning process, the time period used, and how these risks and opportunities are priori-

tized. The report does not describe the impact of climate-related issues on their operations, supply chain, and investment in research and development. The report does not describe their plans for transitioning to a low-carbon economy.

**Score4: 0**

Q5: How resilient is the organisation's strategy when considering different climate-related scenarios, including a 2°C target or lower scenario? How resilient is the organisation's strategy when considering climate physical risks?

A5: The report does not provide sufficient information to evaluate the resilience of JPMorgan Chase's strategy in different climate-related scenarios or physical risks. The report mainly focuses on the company's social and economic initiatives, with little mention of climate risks or strategies to mitigate them.

Analysis5: The sustainability report provided no information on the resilience of the organization's strategy to climate-related risks and opportunities, including a transition to a low-carbon economy consistent with a 2°C or lower scenario and scenarios consistent with increased physical climate-related risks. The report did not discuss how the organization's strategies might change to address potential risks and opportunities, the potential impact of climate-related issues on financial performance and financial position, or the climate-related scenarios and associated time horizon considered. Therefore, the report failed to meet any of the requirements for high-quality disclosure pertaining to the critical element of the organization's strategy's resilience to climate-related risks and opportunities.

**Score5: 0**

**Risk management:**

Q6: What processes does the organisation use to identify and assess climate-related risks?

A6: The report does not provide specific information on the processes used by JPMorgan Chase to identify and assess climate-related risks. The report only mentions the company's commitment to addressing climate change and its efforts to reduce its carbon footprint.

Analysis6: The sustainability report provided no information on the processes used by JPMorgan Chase to identify and assess climate-related risks. The report did not describe the organization's risk



management processes for identifying and assessing climate-related risks, nor did it provide any information on how the organization determines the relative significance of climate-related risks in relation to other risks. The report did not disclose whether the organization considers existing and emerging regulatory requirements related to climate change or other relevant factors. The report did not provide any information on the processes for assessing the potential size and scope of identified climate-related risks or definitions of risk terminology used or references to existing risk classification frameworks used. As a result, the report does not meet any of the requirements for high-quality disclosure pertaining to the critical element of identifying and assessing climate-related risks.

**Score6: 0**

Q7: How does the organisation manage climate-related risks?

A7: JPMorgan Chase's sustainability report does not provide concrete information on how the organization manages climate-related risks. The report only mentions governance issues within the business, but no specific actions or strategies are outlined to address climate risks.

Analysis7: The given disclosure does not provide any information on how JPMorgan Chase manages climate-related risks. The report does not describe the organization's processes for managing climate-related risks, including how they make decisions to mitigate, transfer, accept, or control those risks. Additionally, the report does not describe their processes for prioritizing climate-related risks, including how materiality determinations are made within their organizations. The disclosure is irrelevant and does not meet any of the requirements for high-quality disclosure pertaining to the critical element of how the organization manages climate-related risks.

**Score7: 0**

Q8: How are the processes for identifying, assessing, and managing climate-related risks integrated into the organisation's overall risk management?

A8: The report does not provide sufficient information on how the processes for identifying, assessing, and managing climate-related risks are integrated into the organization's overall risk management. The report only mentions the company's commitment to sustainability and its efforts to reduce its

carbon footprint, but it lacks concrete data and specific actions taken to address climate-related risks.

Analysis8: The given sustainability report does not provide any information on how the processes for identifying, assessing, and managing climate-related risks are integrated into the organization's overall risk management. The report contains irrelevant and unrelated content that does not address the TCFD recommendation point. Therefore, the report fails to meet the TCFD guidelines for high-quality disclosure on the TCFD recommendation point.

**Score8: 0**

**Metrics and targets:**

Q9: What metrics does the organisation use to assess climate-related risks and opportunities? How do the metrics help ensure that the performance is in line with its strategy and risk management process?

A9: The report does not provide specific metrics used by JPMorgan Chase to assess climate-related risks and opportunities. It only mentions that the company considers climate change as a significant risk and opportunity and has integrated it into its risk management process and business strategy. However, the report lacks concrete data to support these claims.

Analysis9: The given disclosure does not provide any information on the metrics used by the organization to assess climate-related risks and opportunities. Instead, it contains general information about the company's economic growth, partnerships, and initiatives. The disclosure does not meet any of the requirements outlined for this critical element. There is no mention of key metrics used to measure and manage climate-related risks and opportunities, metrics consistent with the cross-industry, metrics on climate-related risks associated with water, energy, land use, and waste management, or internal carbon prices. The disclosure also does not provide any information on how related performance metrics are incorporated into remuneration policies or climate-related opportunity metrics. Therefore, the score for this disclosure is 0.

**Score9: 0**

Q10: Does the organisation disclose its Scope 1, Scope 2, and, if appropriate, Scope 3 greenhouse gas (GHG) emissions? What are the related risks

and do they differ depending on the scope?

A10: JPMorgan Chase does disclose its Scope 1 and Scope 2 GHG emissions, but not its Scope 3 emissions. The report does not provide any specific figures on these emissions. The related risks are not mentioned in the report.

Analysis10: The sustainability report of JPMorgan Chase does not provide any information on the organization's Scope 1, Scope 2, and Scope 3 greenhouse gas emissions. The report does not meet any of the requirements for high-quality disclosure pertaining to the critical element. The report does not provide any information on the related risks and how they differ depending on the scope. The report does not provide any information on the methodologies used to calculate or estimate the metrics. The report does not provide any historical data to allow for trend analysis. The report does not provide any industry-specific GHG efficiency ratios. The report does not consider disclosing Scope 3 GHG emissions.

**Score10: 0**

Q11: What targets does the organisation use to understand/quantify/benchmark climate-related risks and opportunities? How is the organization performing against these targets?

A11: The report does not provide any specific targets used by JPMorgan Chase to understand/quantify/benchmark climate-related risks and opportunities. Therefore, it is impossible to evaluate the organization's performance against these targets.

Analysis11: The sustainability report provided no information on the targets used by the organization to manage climate-related risks and opportunities and performance against targets. The report did not describe any key climate-related targets such as those related to GHG emissions, water usage, energy usage, etc. The report did not provide any information on whether the target is absolute or intensity based, time frames over which the target applies, base year from which progress is measured, and key performance indicators used to assess progress against targets. The report did not disclose associated interim targets in aggregate or by business line, where available. The methodologies used to calculate targets and measures were not described. Therefore, the report does not meet any of the requirements for high-quality disclosure pertaining to the critical element.

**Score11: 0**

**Average score: 0.0**

## H.2 JP Morgan Chase 2021

### Governance:

Q1: How does the company's board oversee climate-related risks and opportunities?

A1: The board oversees climate-related risks and opportunities through the Public Responsibility Committee, which oversees reputational and conduct risks, within its scope of responsibility, and the Risk Committee, which is responsible for setting the internal approach to managing climate risk, including developing relevant policies and standards as well as building the approach to climate risk data. The board also receives periodic updates on ESG initiatives from senior leaders.

Analysis1: The sustainability report provides some information on the board's oversight of climate-related risks and opportunities. It mentions the existence of a dedicated Climate Risk team and the development of relevant policies and standards. The report also states that the board oversees ESG matters and receives periodic updates on ESG initiatives. However, the report lacks specific details on the processes and frequency by which the board is informed about climate-related issues, whether the board considers climate-related issues when reviewing and guiding strategy, and how the board monitors and oversees progress against goals and targets for addressing climate-related issues. Overall, the disclosure is insufficient to fully satisfy the TCFD guidelines.

**Score1: 60**

Q2: What is the role of management in assessing and managing climate-related risks and opportunities?

A2: JPMorgan Chase has a dedicated Climate Risk team that establishes the internal approach to managing climate risk, develops relevant policies and standards, and integrates climate considerations into line-of-business risk management frameworks. The team collaborates with stakeholders across the Firm to prepare for the impacts of climate change on locations, people, technology, and suppliers. The company plans to publish a standalone climate report later this year.

Analysis2: The disclosure provides some infor-

mation on management's role in assessing and managing climate-related risks and opportunities. It mentions the existence of a dedicated Climate Risk team and an internal risk classification system. However, it lacks specific details on the organizational structure, processes for informing management about climate-related issues, and how management monitors climate-related issues. The disclosure also refers to a standalone climate report that will provide additional detail on the efforts to integrate climate risk into the Firmwide risk management framework. Overall, the disclosure partially meets some of the TCFD guidelines, but it lacks specific details and does not provide a comprehensive overview of management's role in assessing and managing climate-related risks and opportunities.

**Score2: 60**

**Strategy:**

Q3: What are the most relevant climate-related risks and opportunities that the organisation has identified over the short, medium, and long term? Are risks clearly associated with a horizon?

A3: JPMorgan Chase has identified physical and transition risks as the most relevant climate-related risks. The impacts of these risks are assessed against strategic, market, operational, and credit and investment risk. The company plans to disclose additional detail on its efforts to integrate climate risk into its Firmwide risk management framework in a standalone climate report later this year.

Analysis3: The sustainability report provides some information on the climate-related risks and opportunities that the organization has identified over the short, medium, and long term. The report mentions the specific climate-related issues that could have a material financial impact on the organization and the process used to determine which risks and opportunities could have a material financial impact on the organization. However, the report lacks a clear description of the relevant short-, medium-, and long-term time horizons, taking into consideration the useful life of the organization's assets or infrastructure. The report also does not provide a description of the risks and opportunities by sector and/or geography. Overall, the disclosure quality is moderate.

**Score3: 70**

Q4: How do climate-related risks and opportunities impact the organisation's businesses strategy, economic and financial performance, and financial planning?

A4: JPMorgan Chase is committed to understanding how climate change may influence the risks it manages. The firm has a dedicated Climate Risk team that establishes their internal approach to managing climate risk, including developing relevant policies and standards as well as building their approach to climate risk data. They plan to disclose additional detail on their efforts to integrate climate risk into their Firmwide risk management framework in a standalone climate report which they plan to publish later this year.

Analysis4: The sustainability report provides some information on how climate-related risks and opportunities impact the organization's businesses strategy, economic and financial performance, and financial planning. The report discusses the potential impacts of climate risks on the organization's businesses, strategy, and financial planning, and how these risks are managed across different risk types. The report also mentions the development of an internal risk classification system and a dedicated Climate Risk team. However, the report lacks specific details on how climate-related issues serve as an input to the financial planning process, the impact of climate-related issues on the organization's financial performance and position, and the organization's plans for transitioning to a low-carbon economy. Overall, the report partially meets the TCFD guidelines, and the is 60.

**Score4: 60**

Q5: How resilient is the organisation's strategy when considering different climate-related scenarios, including a 2°C target or lower scenario? How resilient is the organisation's strategy when considering climate physical risks?

A5: JPMorgan Chase has developed an internal risk classification system that projects how various climate-risk drivers can translate into potential impacts for their clients, customers, and firm. They plan to disclose additional detail on their efforts to integrate climate risk into their Firmwide risk management framework in a standalone climate report which they plan to publish later this year. However, the report lacks concrete data on the resilience of their strategy in different climate-related scenarios.

Analysis5: The sustainability report provides some information on the organization's approach to climate-related risks and opportunities, including physical and transition risks. The report mentions the development of an internal risk classification system that projects how various climate-risk drivers can translate into potential impacts for the organization's clients, customers, and the firm. The report also mentions the establishment of a dedicated Climate Risk team responsible for managing climate risk and integrating climate considerations into line-of-business risk management frameworks. However, the report lacks specific details on how the organization's strategy might change to address potential risks and opportunities, the potential impact of climate-related issues on financial performance and financial position, and the climate-related scenarios and associated time horizon considered. The report also mentions plans to disclose additional detail on efforts to integrate climate risk into the firm's risk management framework in a standalone climate report later this year.

**Score5: 70**

#### **Risk management:**

Q6: What processes does the organisation use to identify and assess climate-related risks?

A6: JPMorgan Chase has developed an internal risk classification system that evaluates how their businesses and clients could be impacted by climate change. They have a dedicated Climate Risk team that collaborates with stakeholders across the firm to assess the impacts of climate risks on the four major risk types they manage. They plan to disclose additional detail on their efforts to integrate climate risk into their Firmwide risk management framework in a standalone climate report later this year .

Analysis6: The sustainability report provides some information on the organization's processes for identifying and assessing climate-related risks. The report mentions that the organization has developed an internal risk classification system and a dedicated Climate Risk team that collaborates with stakeholders across the Firm. However, the report lacks specific details on how the organization determines the relative significance of climate-related risks in relation to other risks and whether they consider existing and emerging regulatory requirements related to climate change. The report also

does not disclose the processes for assessing the potential size and scope of identified climate-related risks and definitions of risk terminology used or references to existing risk classification frameworks used. Overall, the disclosure is insufficient in meeting the TCFD guidelines.

**Score6: 50**

Q7: How does the organisation manage climate-related risks?

A7: JPMorgan Chase has a dedicated Climate Risk team that establishes internal policies and standards for managing climate risk. They have developed an internal risk classification system that projects how climate risks could materialize across the four major risk types they manage. They plan to disclose additional detail on their efforts to integrate climate risk into their Firmwide risk management framework in a standalone climate report later this year.

Analysis7: The disclosure provides a comprehensive overview of the organization's approach to managing climate-related risks. The report describes the organization's processes for managing climate-related risks, including how they make decisions to mitigate, transfer, accept, or control those risks. The report also describes the organization's processes for prioritizing climate-related risks, including how materiality determinations are made within their organizations. The report mentions the development of an internal risk classification system that considers the horizon, business sector, and geography to better understand how these risks may emerge within the firm. The disclosure also mentions the establishment of a dedicated Climate Risk team, which collaborates with stakeholders across the firm to integrate climate considerations into line-of-business risk management frameworks. However, the report does not provide specific details on how the organization prioritizes climate-related risks.

**Score7: 90**

Q8: How are the processes for identifying, assessing, and managing climate-related risks integrated into the organisation's overall risk management?

A8: JPMorgan Chase has developed an internal risk classification system and a dedicated Climate Risk team to manage climate-related risks. They plan to disclose additional detail on their efforts to integrate climate risk into their Firmwide risk

management framework in a standalone climate report which they plan to publish later this year. However, the report lacks concrete data on how climate-related risks are integrated into the organization's overall risk management.

Analysis8: The disclosure provides some information on how the organization integrates climate-related risks into its overall risk management. The report mentions that the organization has developed an internal risk classification system and a dedicated Climate Risk team that collaborates with stakeholders across the Firm. However, the report lacks specific details on how the organization identifies, assesses, and manages climate-related risks. The report also mentions that the organization plans to disclose additional detail on its efforts to integrate climate risk into its Firmwide risk management framework in a standalone climate report, which is yet to be published. Overall, the disclosure provides some information on the integration of climate-related risks into the organization's overall risk management, but lacks specific details and completeness.

**Score8: 70**

#### **Metrics and targets:**

Q9: What metrics does the organisation use to assess climate-related risks and opportunities? How do the metrics help ensure that the performance is in line with its strategy and risk management process?

A9: JPMorgan Chase has developed an internal risk classification system that projects how various climate-risk drivers can translate into potential impacts for clients, customers, and the firm. The firm has a dedicated Climate Risk team that collaborates with stakeholders across the firm to integrate climate considerations into line-of-business risk management frameworks. The firm plans to disclose additional detail on its efforts to integrate climate risk into its firm-wide risk management framework in a standalone climate report which it plans to publish later this year.

Analysis9: The sustainability report provides some information on the metrics used to assess climate-related risks and opportunities. The report mentions that the organization has developed an internal risk classification system and a dedicated Climate Risk team to manage climate risk. However, the report lacks specific details on the metrics used to

measure and manage climate-related risks and opportunities. The report does not provide metrics on climate-related risks associated with water, energy, land use, and waste management. The report also does not mention whether and how related performance metrics are incorporated into remuneration policies. While the report mentions that the organization plans to disclose additional detail on its efforts to integrate climate risk into its Firmwide risk management framework in a standalone climate report, it does not provide forward-looking metrics for the cross-industry or a description of the methodologies used to calculate or estimate climate-related metrics.

**Score9: 50**

Q10: Does the organisation disclose its Scope 1, Scope 2, and, if appropriate, Scope 3 greenhouse gas (GHG) emissions? What are the related risks and do they differ depending on the scope?

A10: JPMorgan Chase discloses its Scope 1, Scope 2, and Scope 3 greenhouse gas emissions. The majority of their business travel-related emissions are Scope 3 emissions from commercially operated air and rail, reimbursed personal vehicle and rental car travel, and hotel stays. The risks associated with these emissions are not specified in the report.

Analysis10: The sustainability report provides disclosure on Scope 1, Scope 2, and Scope 3 GHG emissions, and the related risks. The report also includes GHG emissions intensity and historical data for trend analysis. However, it is unclear whether the GHG emissions have been calculated in line with the GHG Protocol methodology, and whether the organization has provided generally accepted industry-specific GHG efficiency ratios. The report meets the requirement of disclosing Scope 3 GHG emissions, but it is not clear whether the organization has provided a description of the methodologies used to calculate or estimate the metrics. Overall, the report provides some information on the TCFD recommendation point, but there is room for improvement in terms of providing more specific details.

**Score10: 75**

Q11: What targets does the organisation use to understand/quantify/benchmark climate-related risks and opportunities? How is the organization performing against these targets?

A11: JPMorgan Chase plans to disclose additional

detail on its efforts to integrate climate risk into its Firmwide risk management framework in a standalone climate report which it plans to publish later this year. No specific targets were mentioned in the report.

Analysis11: The sustainability report provides limited information on the targets used by the organization to manage climate-related risks and opportunities and performance against targets. While the report mentions the development of relevant policies and standards, it does not provide specific details on the key climate-related targets, time frames, base year, or key performance indicators used to assess progress against targets. The report also lacks information on the methodologies used to calculate targets and measures. While the report mentions the establishment of a robust governance process for tracking progress, it does not provide specific details on the process. Overall, the report does not meet the TCFD guidelines for high-quality disclosure pertaining to the TCFD recommendation point.

**Score11: 20**

**Average score: 61.36**

### H.3 Shell 2022

#### Governance:

Q1: How does the company's board oversee climate-related risks and opportunities?

A1: The board oversees climate-related risks and opportunities through the Shell Energy and Sustainability Committee, which reviews policies, practices, targets, and performance of Shell, primarily with respect to safety, environment including climate change, and broader sustainability. The committee also reviews wider matters of public concern such as plastic waste, methane emissions, the flaring of natural gas, water scarcity, just transition, human rights, diversity and inclusion, and access to energy in low- and middle-income countries. However, the report lacks concrete data on the board's direct actions and responsibilities.

Analysis1: The disclosure provided by the company does not adequately address the requirements for describing the board's oversight of climate-related risks and opportunities. While the report mentions the existence of a committee that reviews policies, practices, targets, and performance of the

company with respect to climate change, it does not provide any details on how the board is informed about climate-related issues, how it considers climate-related issues when reviewing and guiding strategy, risk management policies, and business plans, and how it monitors and oversees progress against goals and targets for addressing climate-related issues. The report provides some information on the company's climate-related financial disclosures and its membership in various organizations, but this information is not sufficient to meet the requirements for high-quality disclosure.

**Score1: 20**

Q2: What is the role of management in assessing and managing climate-related risks and opportunities?

A2: Shell's management is responsible for assessing and managing climate-related risks and opportunities. They have established internal carbon performance standards or industry benchmarks for projects expected to have a material greenhouse gas impact. However, the report lacks concrete data on how they are managing physical risks and the potential impact of climate change on their operations.

Analysis2: The disclosure provides some information on management's role in assessing and managing climate-related risks and opportunities. It mentions that projects under development that are expected to have a material greenhouse gas impact must meet internal carbon performance standards or industry benchmarks. It also identifies climate change and the associated energy transition as a material risk and mentions that Shell has established remuneration policies designed to support short-term climate targets. However, the disclosure lacks specific information on whether the organization has assigned climate-related responsibilities to management-level positions or committees, the associated organizational structure, processes by which management is informed about climate-related issues, and how management monitors climate-related issues. Therefore, the disclosure is insufficient to meet the TCFD guidelines.

**Score2: 40**

#### Strategy:

Q3: What are the most relevant climate-related

risks and opportunities that the organisation has identified over the short, medium, and long term? Are risks clearly associated with a horizon?

A3: Shell has identified climate change and the associated energy transition as a material risk. The risk could potentially result in changes to the demand for their products, operational costs, supply chains, markets, the regulatory environment, their license to operate, and litigation. They have established remuneration policies designed to support them in achieving their short-term climate targets. Shell has also identified physical risks associated with climate change and takes climate variability into consideration in the design and operation of their assets and infrastructure to minimize the risk of adverse incidents to their employees and contractors, the communities where they operate, their equipment, and infrastructure. Shell has established internal carbon performance standards or industry benchmarks for projects under development that are expected to have a material greenhouse gas impact. They have identified various climate-related risks and opportunities, including energy affordability and accessibility, environmental, social, and governance risks involved in the transition to renewable solutions, offsets, stakeholder engagements, and involvement with the Science Based Targets initiative's social aspects. However, the report lacks concrete data and cross-referencing of topics, and some statements may be cheap talks.

Analysis3: The sustainability report provides some information on climate-related risks and opportunities, but it falls short of meeting the TCFD guidelines for high-quality disclosure. The report does not provide a clear description of the relevant short-, medium-, and long-term time horizons, nor does it describe the specific climate-related issues that could have a material financial impact on the organization. The report does mention the process used to determine which risks and opportunities could have a material financial impact on the organization, but it does not provide a sector or geography-wise breakdown. Overall, the report lacks specificity and detail, making it difficult to assess the organization's climate-related risks and opportunities.

**Score3: 40**

Q4: How do climate-related risks and opportunities impact the organisation's businesses strategy, economic and financial performance, and financial

planning?

A4: Shell identifies climate change and the energy transition as material risks that could affect demand, operational costs, supply chains, markets, regulatory environment, license to operate, and litigation. The company has established internal carbon performance standards for projects with material greenhouse gas impact and has set short-term climate targets. However, the report lacks concrete data on the financial impact of climate-related risks and opportunities on the company's strategy and financial performance.

Analysis4: The sustainability report provides some information on how climate-related risks and opportunities impact the organization's businesses strategy, economic and financial performance, and financial planning. The report discusses how climate-related risks are assessed at a project level and how they affect the demand for products, operational costs, supply chains, markets, the regulatory environment, license to operate, and litigation. However, the report lacks specific details on the impact of climate-related issues on the organization's financial performance and financial position. The report also does not describe how climate-related issues serve as an input to their financial planning process, the time period used, and how these risks and opportunities are prioritized. Overall, the report partially meets the TCFD guidelines for high-quality disclosure pertaining to the TCFD recommendation point.

**Score4: 60**

Q5: How resilient is the organisation's strategy when considering different climate-related scenarios, including a 2°C target or lower scenario? How resilient is the organisation's strategy when considering climate physical risks?

A5: Shell has identified climate change and the associated energy transition as a material risk. The company has established internal carbon performance standards for projects under development that are expected to have a material greenhouse gas impact. Shell's climate target is to become a net-zero emissions energy business by 2050. However, the report lacks concrete data on how resilient the organization's strategy is when considering different climate-related scenarios or physical risks.

Analysis5: The sustainability report provides some information on how Shell's strategy is resilient to climate-related risks and opportunities, but it falls

short of meeting the TCFD guidelines outlined for this TCFD recommendation point. The report mentions that assessing climate-related risks is important for making investment decisions and that Shell has identified climate change as a material risk. However, it does not provide specific details on how the organization's strategy might change to address potential risks and opportunities, the potential impact of climate-related issues on financial performance and position, or the climate-related scenarios and associated time horizons considered. The report also lacks information on how resilient Shell's strategy is when considering physical climate risks. Overall, the report provides insufficient information to assess the quality of disclosure on this TCFD recommendation point.

**Score5: 40**

#### **Risk management:**

Q6: What processes does the organisation use to identify and assess climate-related risks?

A6: Shell assesses climate-related risks at a project level by ensuring that projects meet internal carbon performance standards or industry benchmarks. The company has identified climate change as a material risk that could affect its demand, operational costs, supply chains, markets, regulatory environment, license to operate, and litigation. Shell also considers physical risks associated with climate change in the design and operation of its assets and infrastructure. The company is a member of the Taskforce on Nature-related Financial Disclosures Forum and joined the Science Based Targets Network Corporate Engagement Program in 2022 to develop a risk management and disclosure framework for organizations to report and act on evolving nature-related risks.

Analysis6: The sustainability report provides some information on the organization's processes for identifying and assessing climate-related risks. The report mentions that assessing climate-related risks is an important part of making initial investment decisions and that the organization has established internal carbon performance standards or industry benchmarks. However, the report lacks specific details on how the organization determines the relative significance of climate-related risks in relation to other risks, whether they consider existing and emerging regulatory requirements related to climate change, and the processes for assessing

the potential size and scope of identified climate-related risks. The report also does not provide definitions of risk terminology used or references to existing risk classification frameworks used. Overall, the disclosure is insufficient to meet the TCFD guidelines.

**Score6: 40**

Q7: How does the organisation manage climate-related risks?

A7: Shell manages climate-related risks by assessing them at a project level and ensuring that projects meet internal carbon performance standards or industry benchmarks. They also take climate variability into consideration in the design and operation of their assets and infrastructure. Shell has identified climate change as a material risk that could result in changes to demand, operational costs, supply chains, markets, regulatory environment, license to operate, and litigation. They have established remuneration policies to support short-term climate targets and are a member of the Taskforce on Nature-related Financial Disclosures Forum. Shell invests in nature-based solution projects that follow the Climate, Community and Biodiversity Standards. They have set targets to reduce their absolute Scope 1 and 2 emissions by 50

Analysis7: The report provides some information on how the organization manages climate-related risks. It describes the processes for managing climate-related risks, including how they make decisions to mitigate, transfer, accept, or control those risks. It also describes the processes for prioritizing climate-related risks, including how materiality determinations are made within their organizations. However, the report lacks specific details on how the organization manages climate-related risks. For instance, it does not provide information on how the organization identifies and assesses climate-related risks, how it monitors and reports on these risks, and how it engages with stakeholders on these risks. Overall, the report provides some information on how the organization manages climate-related risks, but it could be more detailed and specific.

**Score7: 70**

Q8: How are the processes for identifying, assessing, and managing climate-related risks integrated into the organisation's overall risk management?

A8: Shell integrates the assessment of climate-



related risks into its investment decisions by ensuring that projects meet internal carbon performance standards or industry benchmarks. The company also has remuneration policies designed to support short-term climate targets. Shell identifies climate change and the associated energy transition as a material risk that could result in changes to the demand for its products, operational costs, supply chains, markets, regulatory environment, license to operate, and litigation. The company also takes climate variability into consideration in the design and operation of its assets and infrastructure to minimize the risk of adverse incidents to its employees and contractors, the communities where it operates, its equipment, and infrastructure.

Analysis8: The sustainability report provides some information on how the organization integrates processes for identifying, assessing, and managing climate-related risks into its overall risk management. The report mentions that assessing climate-related risks is an important part of making initial investment decisions, and projects under development that are expected to have a material greenhouse gas impact must meet internal carbon performance standards or industry benchmarks. The report also discusses the potential physical impacts of climate change and how the organization takes climate variability into consideration in the design and operation of its assets and infrastructure. However, the report lacks specific details on how the organization identifies and assesses climate-related risks and how it manages these risks. Therefore, the report only partially meets the TCFD guidelines.

**Score8: 60**

### **Metrics and targets:**

Q9: What metrics does the organisation use to assess climate-related risks and opportunities? How do the metrics help ensure that the performance is in line with its strategy and risk management process?

A9: Shell uses internal carbon performance standards or industry benchmarks to assess climate-related risks at a project level. The standards are used to measure a project's average lifetime greenhouse gas intensity or energy efficiency per asset type. The company has also established remuneration policies to support short-term climate targets. The 2022 Annual Report provides more information on material climate-related risks and opportu-

nities.

Analysis9: The sustainability report provides some information on the metrics used to assess climate-related risks and opportunities. The report mentions internal carbon performance standards and industry benchmarks used to measure and manage climate-related risks and opportunities. However, the report lacks specific details on the metrics used to assess climate-related risks and opportunities. The report does not provide metrics consistent with the cross-industry, metrics on climate-related risks associated with water, energy, land use, and waste management, or forward-looking metrics for the cross-industry. The report does not describe whether and how related performance metrics are incorporated into remuneration policies. The report provides some information on internal carbon prices and climate-related opportunity metrics. Overall, the report lacks specific details on the metrics used to assess climate-related risks and opportunities, and therefore, the score is 50.

**Score9: 50**

Q10: Does the organisation disclose its Scope 1, Scope 2, and, if appropriate, Scope 3 greenhouse gas (GHG) emissions? What are the related risks and do they differ depending on the scope?

A10: Yes, Shell discloses its Scope 1 and 2 greenhouse gas emissions. Scope 3 emissions from energy products sold account for most of the total emissions reported. The related risks include regulatory and reputational risks, as well as risks associated with the energy transition. The report lacks specific information on the risks associated with each scope.

Analysis10: The sustainability report provides a comprehensive disclosure of Scope 1 and Scope 2 GHG emissions under operational control, including a breakdown of emissions by gas type and historical trends. However, the report lacks a clear disclosure of Scope 3 GHG emissions and associated risks, which are critical for a complete understanding of the company's carbon footprint. The report does mention that Scope 3 emissions from energy products sold account for most of the total emissions, but it does not provide any further details. The report also lacks a clear description of the methodologies used to calculate or estimate the metrics. Overall, the report partially meets the requirements for high-quality disclosure pertaining to the critical element of GHG emissions disclosure.

**Score10: 70**

Q11: What targets does the organisation use to understand/quantify/benchmark climate-related risks and opportunities? How is the organization performing against these targets?

A11: Shell uses internal carbon performance standards or industry benchmarks to assess climate-related risks at a project level. The company aims to reduce absolute Scope 1 and 2 emissions by 50

Analysis11: The report provides some information on the targets used by the organization to manage climate-related risks and opportunities and performance against targets. The report mentions Shell's climate target to become a net-zero emissions energy business by 2050 and includes some specific targets such as reducing absolute Scope 1 and 2 emissions by 50

**Score11: 60**

**Average score: 50.00**

#### H.4 UBS 2022

##### Governance:

Q1: How does the company's board oversee climate-related risks and opportunities?

A1: The Board of Directors' Corporate Culture and Responsibility Committee oversees the company's climate risk program and approves Group-wide sustainability and impact objectives. It considers the progress of UBS's climate risk program jointly with the BoD Risk Committee and supports the BoD in fulfilling its duty to set and supervise an appropriate risk management and control framework.

Analysis1: The disclosure provides some information on the board's oversight of climate-related risks and opportunities. It mentions that the board is responsible for monitoring climate-related risks and that designated management positions ensure the development, implementation, and review of policies and metrics. The board also considers progress against goals and targets for addressing climate-related issues. However, the disclosure lacks specific details on the processes and frequency by which the board and/or board committees are informed about climate-related issues. It also does not provide sufficient information on whether the board and/or board committees consider climate-related issues when reviewing and

guiding strategy, major plans of action, risk management policies, annual budgets, and business plans. Overall, the disclosure partially meets the requirements for high-quality disclosure pertaining to the board's oversight of climate-related risks and opportunities.

**Score1: 60**

Q2: What is the role of management in assessing and managing climate-related risks and opportunities?

A2: UBS AG integrates climate-related risks into its investment and risk management processes. Designated management-level positions ensure the development, implementation, and review of framework, policies, and metrics. The company applies a sustainability and climate risk policy framework to all relevant activities. However, the report lacks concrete data and relies on vague statements, and it is unclear how the company is addressing the lack of data for some portfolios.

Analysis2: The disclosure provides some information on management's role in assessing and managing climate-related risks and opportunities. It mentions that regular management meetings are held to monitor the status and progress of efforts to manage climate-related risks. It also describes the responsibilities of designated management and the processes by which management is informed about climate-related issues. However, the disclosure lacks specific details on the organizational structure and how management monitors climate-related issues. It also does not mention whether the management positions or committees report to the board or a committee of the board. Overall, the disclosure partially meets the requirements for high-quality disclosure pertaining to the critical element.

**Score2: 60**

##### Strategy:

Q3: What are the most relevant climate-related risks and opportunities that the organisation has identified over the short, medium, and long term? Are risks clearly associated with a horizon?

A3: UBS AG has identified climate-driven risks and opportunities across different time horizons. Physical risks are moderately low, while transition risks are emerging in the long term. Climate-related risks are scored between 0 and 1, based on trans-

mission channels, and are rated from low to high. Climate-related investment products are seen as the highest-ranked immediate commercial opportunity. However, the report lacks concrete data and timelines for risk management and opportunity assessment.

**Analysis3:** The report provides a detailed description of the climate-related risks and opportunities identified by the organization over the short, medium, and long term. The report describes the relevant time horizons, specific climate-related issues, and the process used to determine which risks and opportunities could have a material financial impact on the organization. The report also considers providing a description of risks and opportunities by sector and/or geography, as appropriate. However, the report lacks specific details on the process used to determine the materiality of the risks and opportunities identified. Overall, the report meets most of the requirements, but there is room for improvement in terms of providing more specific details.

**Score3: 85**

**Q4:** How do climate-related risks and opportunities impact the organisation's businesses strategy, economic and financial performance, and financial planning?

**A4:** Climate-related risks represent financial risks for UBS and its clients, while investing in climate change mitigation presents commercial opportunities. UBS has a climate strategy that covers managing climate-related financial risks and taking action on a net-zero future. However, the report lacks concrete data and relies on vague statements, and there is a risk of greenwashing.

**Analysis4:** The sustainability report provides a comprehensive discussion of climate-related risks and opportunities and their impact on the organization's businesses, strategy, and financial planning. The report describes the impact of climate-related issues on various areas, including products and services, supply chain, adaptation and mitigation activities, investment in research and development, operations, acquisitions or divestments, and access to capital. The report also describes how climate-related issues serve as an input to the financial planning process, the time period used, and how these risks and opportunities are prioritized. The report provides a holistic picture of the interdependencies among the factors that affect the organization's

ability to create value over time. However, the report lacks specific details on GHG emissions targets and specific activities intended to reduce GHG emissions in their operations and value chain or to otherwise support the transition.

**Score4: 90**

**Q5:** How resilient is the organisation's strategy when considering different climate-related scenarios, including a 2°C target or lower scenario? How resilient is the organisation's strategy when considering climate physical risks?

**A5:** UBS AG has integrated climate risk in the risk control and monitoring process including scenario analysis. However, for some portfolios, the assessment of climate-related risks is not possible due to lack of data. The company has developed climate- and nature-related risk methodologies, which rate cross-sectoral exposures to SCR sensitivity, on a scale from high to low. The report does not provide enough information to determine the resilience of the organization's strategy when considering different climate-related scenarios or physical risks.

**Analysis5:** The sustainability report provides a detailed description of the organization's methodology for assessing climate-driven risks and how it has integrated climate risk in the risk control and monitoring process, including scenario analysis. The report also discusses the potential impact of climate-related issues on financial performance and position. However, the report lacks specific details on how the organization's strategies might change to address potential risks and opportunities, and the climate-related scenarios and associated time horizon considered. Additionally, the report does not provide a clear description of the organization's resilience to a 2°C or lower scenario. Overall, the report provides a good level of disclosure but lacks some specific details to fully satisfy the TCFD guidelines.

**Score5: 80**

#### **Risk management:**

**Q6:** What processes does the organisation use to identify and assess climate-related risks?

**A6:** UBS AG uses standard financial and non-financial risk processes to identify and assess material sustainability and climate risks. These processes include controls during client onboarding, transaction due diligence, product development, in-

vestment decision processes, own operations, supply chain management, and portfolio reviews. The company also integrates climate risk in the risk control and monitoring process, including scenario analysis, and periodically reports on sustainability and climate risk exposures. However, the report acknowledges that climate risk analysis is a novel area of research, and methodologies, tools, and data availability are still evolving.

**Analysis6:** The disclosure provides some information on the organization's processes for identifying and assessing climate-related risks. The report mentions that standard financial and non-financial risk processes ensure that material sustainability and climate risks are identified, assessed, approved, and escalated in a timely manner. The report also mentions that climate risk is integrated into the risk control and monitoring process, including scenario analysis. However, the report lacks specific details on how the organization determines the relative significance of climate-related risks in relation to other risks. The report also does not disclose the processes for assessing the potential size and scope of identified climate-related risks and definitions of risk terminology used or references to existing risk classification frameworks used. Therefore, the report partially meets the requirements, and the score is 60.

**Score6: 60**

**Q7:** How does the organisation manage climate-related risks?

**A7:** UBS AG integrates risk data and insights into their investment management processes to manage climate risks. They have a sustainability and climate risk management framework in place, which includes risk identification and measurement, monitoring and risk appetite setting, risk management and control, and risk reporting processes. They also offer innovative products and services to help clients manage and protect their assets from climate-related risks.

**Analysis7:** The disclosure provides a comprehensive overview of the organization's approach to managing climate-related risks. The report describes the processes for managing climate-related risks, including how they make decisions to mitigate, transfer, accept, or control those risks. The report also describes the processes for prioritizing climate-related risks, including how materiality determinations are made within their organizations.

The report provides details on the risk management and control processes, including the identification, measurement, monitoring, and escalation of material sustainability and climate risks. However, the report could have provided more specific details on the methodologies used to assess climate-related risks and how the organization integrates climate risk data and insights into its investment management processes.

**Score7: 90**

**Q8:** How are the processes for identifying, assessing, and managing climate-related risks integrated into the organisation's overall risk management?

**A8:** UBS AG integrates sustainability and climate risk considerations into its internal and external reporting, risk appetite setting, and management and control processes. The company has a climate risk program with defined responsibilities, processes, and tools applicable to business divisions and group functions. However, the report lacks concrete data on the integration of climate-related risks into the organization's overall risk management.

**Analysis8:** The disclosure provides a comprehensive description of how the organization integrates processes for identifying, assessing, and managing climate-related risks into its overall risk management. The report describes how sustainability and climate risk exposures are monitored and metrics reported internally to enable risk appetite setting. The report also describes how management and control processes ensure that material sustainability and climate risks are identified, measured, monitored, and escalated in a timely manner. However, the report lacks specific details on how the organization integrates climate-related risks into its overall risk management. For instance, the report does not provide specific examples of how sustainability and climate risk considerations are included in internal and external reporting. Overall, the disclosure meets most of the requirements and scores 85 out of 100.

**Score8: 85**

**Metrics and targets:**

**Q9:** What metrics does the organisation use to assess climate-related risks and opportunities? How do the metrics help ensure that the performance is in line with its strategy and risk management process?

A9: UBS AG uses periodic climate-related credit and market risk metrics, climate risk heatmaps, climate scenario analysis and stress test exercises, and a sustainability and climate risk scorecard to assess climate-related risks and opportunities. These metrics help ensure that the performance is in line with its strategy and risk management process by integrating risk data and insights into its investment management processes and enabling risk appetite setting.

Analysis9: The given disclosure provides some information on the metrics used to assess climate-related risks and opportunities. The report mentions that key sustainability and climate risk considerations are included in internal and external reporting, and that sustainability and climate risk exposures are monitored and metrics reported internally to enable risk appetite setting. However, the report lacks specific details on the metrics used to measure and manage climate-related risks and opportunities. The report does not provide metrics on climate-related risks associated with water, energy, land use, and waste management. The report also does not provide forward-looking metrics for the cross-industry, consistent with their business or strategic planning time horizons. Overall, the disclosure is insufficient in meeting the TCFD guidelines.

**Score9: 40**

Q10: Does the organisation disclose its Scope 1, Scope 2, and, if appropriate, Scope 3 greenhouse gas (GHG) emissions? What are the related risks and do they differ depending on the scope?

A10: Yes, UBS AG discloses its Scope 1, Scope 2, and Scope 3 greenhouse gas emissions. The report identifies direct emissions from gas, oil and fuel consumption, indirect emissions from electricity and district heating and other indirect emissions from leased assets, paper, waste and business travel. The report also mentions that they are looking to quantify any relevant scope 3 emissions from categories 1 to 14. The related risks are not explicitly stated.

Analysis10: The sustainability report provides a comprehensive disclosure of Scope 1 and Scope 2 GHG emissions, and a partial disclosure of Scope 3 GHG emissions. The report mentions that the organization has made a first estimation of Scope 3 emissions and identified the top contributors. However, it does not provide a detailed breakdown of

Scope 3 emissions by category, which is a requirement for high-quality disclosure. The report also lacks information on the related risks and how they differ depending on the scope. Overall, the report partially meets the requirements for high-quality disclosure of GHG emissions and related risks, but there is room for improvement.

**Score10: 70**

Q11: What targets does the organisation use to understand/quantify/benchmark climate-related risks and opportunities? How is the organization performing against these targets?

A11: UBS AG uses quantitative climate risk appetite, integrates sustainability and climate risk into policies and processes, builds in-house capacity, centralizes and executes ESG data strategy to understand/quantify/benchmark climate-related risks and opportunities. The report does not provide sufficient information to evaluate the organization's performance against these targets.

Analysis11: The sustainability report provides some information on the targets used by the organization to manage climate-related risks and opportunities and performance against targets. The report mentions that the organization has developed methodologies to set climate-related targets and identify climate-related risks, and that sustainability and climate risk exposures are monitored and metrics reported internally to enable risk appetite setting. However, the report lacks specific details on the key climate-related targets, time frames, base year, and key performance indicators used to assess progress against targets. The report also does not provide a description of the methodologies used to calculate targets and measures. Therefore, while the report provides some information on the TCFD recommendation point, it does not fully satisfy the TCFD guidelines.

**Score11: 50**

**Average score: 70.0**

## **I Comparison with Liu et al. (2023)**

Instead of hallucination-free rates, Liu et al. (2023) measures citation recall and precision to evaluate the attribution quality. Citation recall is similar to our content-dimension hallucination-free rate. It measures the proportion of verification-worthy statements that are fully supported by the associ-

ated citations. However, citation recall is statement-level while our metric is paragraph-level: an answer paragraph is hallucination-free only when all its statements are grounded by evidence. citation precision is the portion of citations that support their statements, while our source-dimension hallucination-free rate measures the proportion of answers that honestly include all supportive sources. It is important to note that Liu et al. (2023) includes queries of various domains to evaluate the trustworthiness of general-purpose search engines, while our human evaluation focus on CHATREPORT.

## J Answers Are Easy to Trace

The following example showcases the traceability of an answer produced by ChatGPT. There are three sentences in the answer. Searching for the underlined keywords, we easily identify the first and second sentence’s evidence sentence in Source 215 and 166 respectively.

|  |
|--|
| <p><b>Answer:</b> Assurant conducted a preliminary scenario analysis in 2021 to identify and analyze climate-related risks faced by 10 of its critical and/or highly vulnerable facilities across the globe. The company's longer-term strategic planning process prioritized climate as a multi-year ESG area of focus. However, the report lacks concrete data on the resilience of the organization's strategy in different climate-related scenarios.</p>        |
| <p><b>Source 215:</b> Preliminary Scenario Analysis: The process of screening for climate-related risks and opportunities across operations and business units informed our preliminary scenario analysis activities. We conducted an initial climate scenario analysis in 2021, with third-party support, to identify and analyze the various climate-related risks faced by 10 of Assurant's critical and/or highly vulnerable facilities across the globe ...</p> |
| <p><b>Source 166:</b> ... Assurant's longer-term strategic planning process, overseen by our Board, prioritized climate as a multi-year ESG area of focus ...</p>  |

## K Answer Comparison

An example of ChatGPT and GPT-4 outputs comparison. ChatGPT answers the question extractively and adheres to the authentic information: asserting that it is “the executive leadership team” that reviews Walmart’s climate strategy at least annually. However, GPT-4 mistakenly summarizes that the board oversees ESG issues through “its Nominating and Governance Committee (NGC)”. It asserts that “the NGC reviews Walmart’s climate strategy at least annually”, which is not accurate. GPT-4 conducts unnecessary summarizations and

inferences. This hallucinates the company’s different committees.

|   |
|---|
| <p><b>Question:</b><br/>How does the company's board oversee climate-related risks and opportunities?</p>   |
| <p><b>ChatGPT Answer:</b><br/>Walmart's board oversees the overall ESG strategy and certain priority issues, including climate-related risks and opportunities. The Chief Sustainability Officer provides updates on the ESG agenda and progress to the Nominating and Governance Committee. The executive leadership team reviews Walmart's climate strategy at least annually. The Nominating and Governance Committee discussed Walmart's ESG strategies and progress, including climate and public policy strategies and practices in 2021.</p> |
| <p><b>GPT-4 Answer:</b><br/>The Walmart Board of Directors oversees the company's climate-related risks and opportunities through its Nominating and Governance Committee, which reviews the climate strategy at least annually. The Board also engages with stakeholders and advocates for an enabling policy environment. The company's executive leadership team and ESG Steering Committee also have oversight responsibility for particular ESG issues.</p>  |

## L TCFD Introduction

The TCFD is an internationally recognized initiative established by the Financial Stability Board in 2015. Its general structure - covering governance, strategy, risk management, and metrics and targets - emerges as the premier candidate of voluntary reporting frameworks to be adopted for regulatory reporting and international standards<sup>8</sup>. The purpose of the TCFD is to provide a voluntary reporting framework to encourage companies to disclose information on climate-related risks and opportunities. This information is crucial for stakeholders and investors to make informed decisions that account for climate-related risks. Companies are asked to disclose this information in their annual reports and financial filings, within their sustainability reports, or as stand-alone documents. To date, most companies have included the information as part of their sustainability reports, due to no or limited assurance and legal liability concerns if it was included in the annual reports. This is why our analysis focuses on content in sustainability reports from the perspective of a reporting standard such as the TCFD, but it can also be extended to any other corporate report. However, it is important to keep in mind that our analysis primarily evalu-

<sup>8</sup>The main emerging standards in this context are the draft disclosure standards provided by the International Financial Reporting Standards' International Sustainability Standards Board (ISSB).

ates the disclosed information rather than directly measuring the genuine implementation of tangible actions by the company.

# RALLE: A Framework for Developing and Evaluating Retrieval-Augmented Large Language Models

Yasuto Hoshi\*, Daisuke Miyashita\*, Youyang Ng, Kento Tatsuno,  
Yasuhiro Morioka, Osamu Torii, Jun Deguchi

Kioxia Corporation, Japan

yasuto1.hoshi@kioxia.com

## Abstract

Retrieval-augmented large language models (R-LLMs) combine pre-trained large language models (LLMs) with information retrieval systems to improve the accuracy of factual question-answering. However, current libraries for building R-LLMs provide high-level abstractions without sufficient transparency for evaluating and optimizing prompts within specific inference processes such as retrieval and generation. To address this gap, we present RALLE, an open-source framework designed to facilitate the development, evaluation, and optimization of R-LLMs for knowledge-intensive tasks. With RALLE, developers can easily develop and evaluate R-LLMs, improving hand-crafted prompts, assessing individual inference processes, and objectively measuring overall system performance quantitatively. By leveraging these features, developers can enhance the performance and accuracy of their R-LLMs in knowledge-intensive generation tasks. We open-source our code at <https://github.com/yhoshi3/RaLLe>.

## 1 Introduction

Large language models (LLMs) have shown great potential for natural language understanding and generation tasks (Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023). However, they face challenges when answering factual questions due to hallucinations (or confabulations) (Bang et al., 2023; Borji, 2023), outdated parametric knowledge (Liska et al., 2022), and memory efficiency of parametric knowledge (e.g., Heinzerling and Inui, 2021). To address these limitations, researchers have turned to the retrieval-augmented approach used in open-domain question answering (QA) (Chen et al., 2017), hereinafter referred to as retrieval-augmented LLMs or *R-LLMs*.

In comparison to closed-book settings where language models generate answers without retrieval,

\* These authors contributed equally to this work.

R-LLMs (open-book settings) enable the retrieval of relevant information from external databases or corpora (Mialon et al., 2023; Ng et al., 2023), which has led to improved accuracy in open-domain QA (Shi et al., 2023). Additionally, R-LLMs can acquire extended features even without additional training, such as explicit references, relief from fact hallucination (Nakano et al., 2021), and easy updates to the knowledge source (e.g., Guu et al., 2020; Ng et al., 2023).

Retrieval-augmented generation needs further research and development to reach its full potential. For example, even though the retriever-reader system has been trained on the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019), its F1 score on the short answer task is 68.3 and still lags behind the oracle F1 score of 75.7 (Asai and Choi, 2021). This implies that further improvements can be made to the retrieval-augmented generation approach. Additionally, users would be probably aware that the outputs generated by R-LLMs may contain factual errors, particularly when applied to knowledge-intensive tasks. However, there is currently a lack of accessible evaluation framework to assess their output quality. This makes it difficult to identify areas for improvement.

Furthermore, having effective tools for developing R-LLMs is crucial. These tools should enable the design of inference steps such as retrieve-then-generate, selecting the combination of retrievers and LLMs, evaluating the performance of the entire system, and testing the prompts used in each inference step. Currently available tools, such as the ChatGPT Retrieval Plugin<sup>1</sup>, Guidance<sup>2</sup>, and LangChain<sup>3</sup> (Chase, 2023), offer a high degree of abstraction, making it challenging to verify the functionality of individual inference steps or opti-

<sup>1</sup><https://github.com/openai/chatgpt-retrieval-plugin>

<sup>2</sup><https://github.com/microsoft/guidance>

<sup>3</sup>Note: Our code does not use either of these.



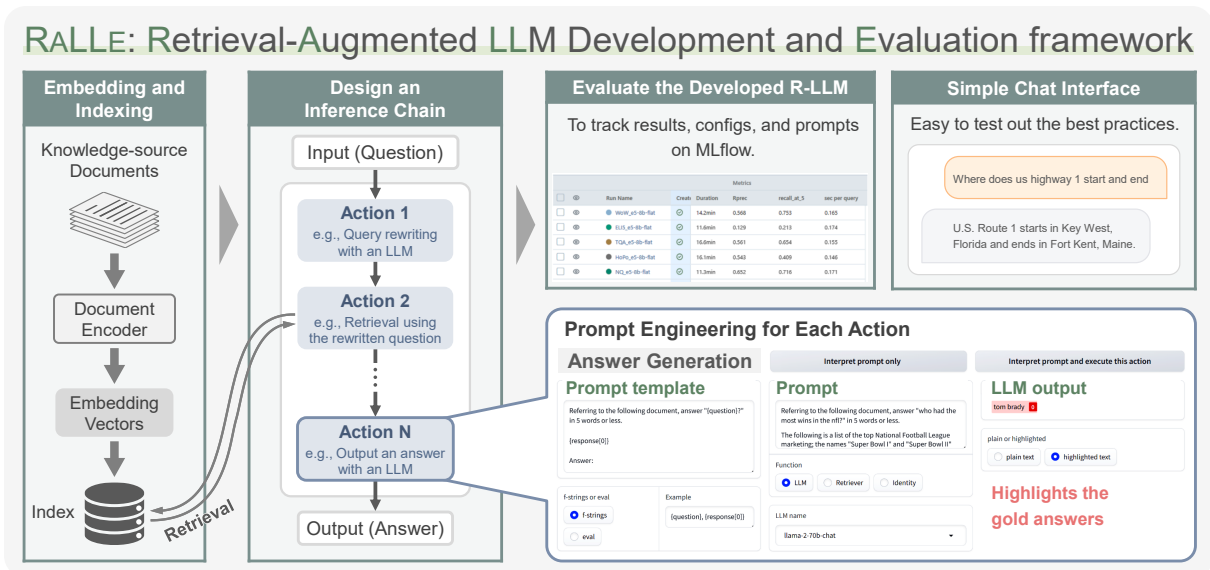


Figure 1: Overview of RALLE, our proposed development and evaluation framework for R-LLMs. Any number of actions can be defined for an R-LLM. Each action can be executed individually to test the corresponding prompts. Experimental setup and evaluation results can be tracked using MLflow. Additionally, a simple chat interface can be built to test out the best practices from the development and evaluation stages in a practical setting.

mize prompts within each step. This lack of transparency might hinder the optimization of R-LLMs.

In this paper, we propose RALLE, an accessible framework for Retrieval-Augmented Large Language model development and Evaluation. We also present evaluation results of several R-LLMs that we have constructed by using open-source retrievers and LLMs. To the best of our knowledge, RALLE is the first framework that empowers R-LLM developers and open-domain QA researchers to efficiently develop, evaluate, and improve R-LLMs using objective metrics.

RALLE offers several key benefits:

1. *Easy development and testing*: users can easily select, combine, and test various retrievers and LLMs, especially open-source models, within a graphical interface.
2. *Objective evaluation of R-LLMs*: RALLE provides reproducible experiments with objective benchmarks/metrics, enabling objective assessments of R-LLM performance.
3. *Transparent prompt engineering*: all inputs (prompts) and outputs of each action are visible to developers, allowing for easy exploration and optimization of the prompts.

## 2 RALLE Usage

Figure 1 presents an overview of the key features of the proposed framework<sup>4</sup>. The primary development process involves three stages: (1) embedding and indexing the knowledge source documents, (2) designing an inference chain consisting of an R-LLM with customized prompt templates for each action, and (3) benchmarking the developed R-LLM.

### 2.1 Document Embedding and Indexing

To begin, the knowledge source documents can be encoded using an arbitrary encoder model, such as a sparse or dense retriever. For efficient indexing of dense embeddings, several methods are available by default, including Faiss (Johnson et al., 2019), HNSW (Malkov and Yashunin, 2020), and DiskANN (Jayaram Subramanya et al., 2019). By default, an HNSW index is constructed with  $ef\_construction = 128$  (the size of the dynamic list for the nearest neighbors) and  $m = 32$  (the number of links created for every new element during graph construction).

### 2.2 Chain Construction

Once the document embedding and indexing are completed, the retrievers (and the corresponding indices) and LLMs can be loaded via the

<sup>4</sup>Please also review the [demonstration screencast](#).

Gradio<sup>5</sup>-based GUI (Abid et al., 2019) to establish an inference chain that comprises an R-LLM. This chain of actions enables users to design a pipeline for multi-step inference, such as `[retrieve]-[generate]`, or more intricate workflows such as `[rewrite query]-[retrieve]-[generate]` proposed in Ma et al. (2023). The versatility of this feature is especially beneficial in creating the chains tailored to specific use cases.

A single-action chain can function as either a simple retriever that returns the retrieved documents, or a *closed-book* QA that leverages the parametric knowledge of an LLM to provide answers without retrieval. In contrast, a chain with multiple actions that include retrieval enables retrieval-augmented generation or *open-book* QA, allowing an LLM to access external documents relevant to a question. Our default setup for R-LLMs consists of two actions: retrieve and generate.

### 2.3 Prompt Engineering

The RALLE framework allows developers to interactively craft customized prompt templates for LLMs and even for search queries on a per-chain basis. Each action can be executed independently, enabling precise control over LLM responses, such as specifying the desired output format or suppressing undesirable hallucinations. To enhance the versatility of prompt development, RALLE integrates support for f-strings and `eval()` function in Python.

### 2.4 Experiment Tracking

We utilize MLflow (LF Projects, 2023) to track the experiments, along with their associated configuration files and prompt templates. This allows us to compare the performance of different experiment runs objectively, which enables us to develop even better R-LLMs.

### 2.5 Chat AI

RALLE also provides support for building a simple chat interface. This enables users to test out best practices from the development and evaluation stages in a practical setting.

## 3 Experimental Settings

In this section, we evaluate the performance of R-LLMs constructed with several combinations of open-source retrievers and LLMs on knowledge-intensive tasks.

<sup>5</sup><https://www.gradio.app/>

### 3.1 Tasks and Datasets

We employ KILT (Knowledge Intensive Language Tasks) benchmark (Petroni et al., 2021), an extensive benchmark that encompasses 11 datasets across five knowledge-intensive natural language processing tasks: fact checking, entity linking, slot filling, open-domain question answering, and dialogue (for further details of KILT, see Petroni et al. (2021)). We use the training sets for developing prompts and the development set for evaluation.

As the knowledge source, we utilize the pre-processed Wikipedia passages provided by KILT. The passages are derived from English Wikipedia articles based on the 2019/08/01 Wikipedia dump data, consisting of a total of 5.9 million articles and 22.2 million 100-word passages. For both dense and sparse retrievers, we use the set of 100-word passages after additional pre-processing that prepends the title of the article to each passage.

Note that RALLE is dataset-agnostic, allowing developers to use their own QA datasets and corpora for development and evaluation. See Appendix A.10 for more information.

### 3.2 Models

This subsection details the retrievers and LLMs employed to build R-LLMs in our experiments. RALLE allows practitioners and researchers to easily experiment with the most recent models available in open-source repositories. With the exception of BM25, all models are available from Hugging Face (Wolf et al., 2020) (see Appendix A.9 for the summary).

#### 3.2.1 LLMs

The LLM used within the R-LLM must comprehend instructions provided in a prompt and generate appropriate responses based on the given information. To achieve this, we use instruction-tuned LLMs with a temperature parameter set to zero for optimal performance and reproducibility.

**Llama-2-chat** is tuned with supervised fine-tuning and reinforcement learning with human feedback (RLHF) (Christiano et al., 2017; Stiennon et al., 2020) to align to human preferences for helpfulness and safety (Touvron et al., 2023b). In our experiments, we utilize both 13-billion (**Llama2-13B**) and 70-billion (**Llama2-70B**) models.

**WizardVicunaLM-13B**<sup>6</sup> (**W-Vicuna-13B**) (Lee, 2023) is formed by combining the concepts

<sup>6</sup><https://huggingface.co/junelee/wizard-vicuna-13b>

| Model | dim.  | max len. | MTEB Retrieval    |
|-------|-------|----------|-------------------|
| BM25  | -     | -        | 42.3 <sup>♠</sup> |
| m-e5  | 1,024 | 514      | 51.43             |
| e5    | 1,024 | 512      | 50.56             |

Table 1: Summary of the retrievers used in our evaluation. Dimensions of a dense embedding vector are shown in *dim.*, while the maximum token length of an input sequence is *max len.*. The evaluation metric for MTEB Retrieval is nDCG@10. <sup>♠</sup>: Results from Ram et al. (2022). Results on MTEB Retrieval except BM25 are copied from MTEB leaderboard<sup>7</sup>.

of WizardLM (Xu et al., 2023) (refining the initial instructions with Evol-Instruct method (Xu et al., 2023)) and Vicuna (Chiang et al., 2023) (a fine-tuned LLaMA model (Touvron et al., 2023a) with multi-round conversation data from chatbots).

### 3.2.2 Retrievers

We experiment with both sparse and dense retrievers for document retrieval. Specifically, we select dense retrievers that have achieved high accuracy on the retrieval task of Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023) leaderboard<sup>7</sup> as of July 2023. A list of the retrievers used in our study can be found in Table 1. In the open-book experiments, the top-5 most relevant documents are retrieved.

As the metrics of retrieval performance, we follow Petroni et al. (2021) and use the page-level R-precision (Craswell, 2016) and recall@5. The page-level R-precision is the percentage of  $R$  gold pages inside each provenance set among the top- $R$  retrieved pages. Typically, R-Precision is equivalent to Precision@1 except FEVER and HotPotQA (multi-hop datasets).

**BM25** (Robertson and Zaragoza, 2009) is a bag-of-words retrieval function based on the term-matching. We use the Pyserini (Lin et al., 2021) implementation of unigram BM25 with the default parameters of  $k_1 = 0.9$  (term frequency scaling) and  $b = 0.4$  (document length normalization). The documents for BM25 retrieval is the same 100-word passages as the dense retrievers.

**e5-large-v2**<sup>8</sup> (**e5**) (Wang et al., 2022) is a supervised bi-encoder model with a query encoder and a

document encoder. **multilingual-e5-large**<sup>9</sup> (**m-e5**) is a multilingual fine-tuned e5 model.

### 3.3 Prompts

We utilize custom-designed prompt templates that are specifically crafted for each dataset in KILT. RALLE accepts templates with non-natural language formats, such as f-strings and eval() functions in Python. This allows developers to carefully craft their prompt templates for optimal performance. The prompt templates used in our experiments are shown in Appendix A.11.

For entity linking task of KILT (AY2, WnWi, and WnCw), we employ a REWRITE-EL template by default for search queries. This template extracts the specific entity mentions being questioned as a query, as employing an entire span of a question is unlikely to find relevant documents (we will discuss in Section 4.3). After retrieving the relevant documents, the top-1 Wikipedia title is output as an answer. As a result, the downstream accuracy in entity linking task is not affected by the number of retrieved documents (if one or more).

## 4 KILT Benchmark Results

This section provides the downstream and retrieval performance of the R-LLMs developed and evaluated using RALLE.

### 4.1 Baseline

We compare our results with those of the BART-large model (Lewis et al., 2020a) for the closed-book setting and the RAG model (Lewis et al., 2020b) for the open-book setting, which presented in Petroni et al. (2021). Notably, these baseline models were specifically fine-tuned on the KILT benchmark, whereas our chosen LLMs and constructed R-LLMs were not. See also Appendix A.5 for additional information of the baselines.

### 4.2 Downstream Performance

We summarize the downstream performance<sup>10</sup> in Table 2. RALLE also includes has\_answer percentage for short answers, a proxy metric to measure the proportion of questions that contain gold answers within the final output generated by an R-LLM (see Appendix A.3 for more details).

<sup>7</sup><https://huggingface.co/spaces/mteb/leaderboard>

<sup>8</sup><https://huggingface.co/intfloat/e5-large-v2>

<sup>9</sup><https://huggingface.co/intfloat/multilingual-e5-large>

<sup>10</sup>See also Table 6 in Appendix A.6 for additional results in a closed-book setting.

| Dataset                               | Fact Check.        | Entity Linking     |                    |                    | Slot Filling       |                    | Open Domain QA     |                    |                    |             | Dial.       |
|---------------------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-------------|-------------|
|                                       | FEV                | AY2                | WnWi               | WnCw               | T-REx              | zsRE               | NQ                 | HoPo               | TQA                | ELI5        | WoW         |
| Model / Metric                        | Accuracy           |                    |                    |                    |                    |                    | Exact Match        |                    |                    | RL          | F1          |
| BART-large <sup>◇</sup> (closed-book) | <u>80.7</u>        | <b>86.6</b>        | 47.9               | <b>48.0</b>        | <u>43.8</u>        | 3.0                | 26.2               | 16.9               | 32.5               | <u>22.7</u> | <b>13.8</b> |
| Llama2-70B (closed-book)              | 33.6 (74.9)        | 39.8 (54.5)        | 42.8 (53.8)        | 39.2 (55.7)        | 28.5 (40.5)        | 11.3 (13.6)        | 19.6 (37.4)        | 13.9 (25.1)        | 67.4 (80.8)        | <b>23.0</b> | <u>13.3</u> |
| RAG <sup>◇</sup>                      | <b>87.7</b>        | <u>77.4</u>        | <b>49.0</b>        | <u>46.7</u>        | <b>61.5</b>        | <b>47.4</b>        | <b>48.8</b>        | <u>27.7</u>        | 61.7               | 16.1        | <u>13.3</u> |
| e5 + W-Vicuna-13B                     | 10.6 (42.4)        | 51.2 (57.9)        | <u>48.6</u> (51.4) | 45.6 (51.4)        | 31.6 (46.1)        | 23.0 (29.3)        | 18.7 (38.0)        | 19.7 (28.3)        | 43.1 (67.7)        | 21.4        | 12.3        |
| e5 + Llama2-13B                       | 66.3 (73.5)        | <u>51.2</u> (57.9) | 48.6 (51.4)        | <u>45.6</u> (51.4) | 17.2 (42.3)        | 31.7 (41.1)        | 36.1 (43.3)        | 14.3 (25.5)        | 56.3 (76.2)        | 20.9        | 12.3        |
| BM25 + Llama2-70B                     | 46.2 (86.3)        | <b>18.0</b> (35.9) | 19.1 (32.2)        | 14.2 (30.9)        | 25.9 (43.0)        | 31.4 (37.8)        | 25.3 (34.3)        | 25.9 (33.4)        | 65.8 (80.0)        | 21.3        | 12.2        |
| e5 + Llama2-70B                       | 49.9 (88.6)        | 51.2 (57.9)        | 48.6 (51.4)        | 45.6 (51.4)        | 28.9 (49.2)        | 35.0 (43.2)        | <u>36.4</u> (48.8) | <b>28.1</b> (35.8) | <u>71.1</u> (83.9) | 21.5        | 13.2        |
| e5 (DiskANN)                          | 49.9 (87.9)        | <b>44.3</b> (50.5) | <b>45.3</b> (48.1) | <b>43.0</b> (48.8) | 25.3 (43.9)        | 32.1 (37.9)        | 36.1 (48.4)        | 26.7 (34.3)        | 70.4 (83.2)        | 21.5        | 13.1        |
| top-2                                 | 49.3 (88.1)        | 51.2 (57.9)        | 48.6 (51.4)        | 45.6 (51.4)        | 23.5 (44.9)        | <b>34.7</b> (43.0) | 33.7 (46.2)        | 23.8 (34.2)        | <b>71.3</b> (82.9) | 21.6        | <u>13.3</u> |
| top-10                                | 50.2 (88.0)        | 51.2 (57.9)        | 48.6 (51.4)        | 45.6 (51.4)        | 31.1 (49.3)        | <u>35.4</u> (42.5) | 35.2 (48.1)        | 24.9 (35.7)        | 59.3 (82.8)        | 21.5        | 13.2        |
| Model / Metric                        | KILT-Accuracy      |                    |                    |                    |                    |                    | KILT-EM            |                    | KILT-RL            | KILT-F1     |             |
| RAG <sup>◇</sup>                      | <b>55.5</b>        | <b>77.4</b>        | <b>49.0</b>        | <b>46.7</b>        | <b>25.4</b>        | <b>42.6</b>        | <b>36.3</b>        | 3.1                | 36.1               | <b>2.7</b>  | 7.5         |
| e5 + W-Vicuna-13B                     | 8.4 (33.5)         | <u>51.2</u> (51.2) | <u>48.6</u> (48.6) | <u>45.5</u> (45.5) | 19.0 (28.0)        | 22.2 (28.1)        | 14.4 (27.8)        | 8.6 (11.9)         | 26.6 (40.3)        | <b>2.7</b>  | 7.3         |
| e5 + Llama2-13B                       | <u>53.1</u> (58.7) | 51.2 (51.2)        | 48.6 (48.6)        | 45.5 (45.5)        | 11.5 (25.7)        | 29.8 (38.5)        | 27.5 (32.5)        | 5.6 (10.6)         | 34.7 (46.1)        | <b>2.7</b>  | 7.4         |
| BM25 + Llama2-70B                     | 21.9 (44.4)        | 17.6 (17.6)        | 18.9 (18.9)        | 13.9 (13.9)        | 14.5 (22.5)        | 24.9 (29.6)        | 9.3 (12.4)         | 4.5 (5.9)          | 23.6 (27.9)        | <u>1.5</u>  | 4.0         |
| e5 + Llama2-70B                       | 40.2 (71.2)        | 51.2 (51.2)        | 48.6 (48.6)        | 45.5 (45.5)        | 19.2 (29.7)        | 32.8 (40.4)        | <u>27.7</u> (36.3) | <b>11.3</b> (14.5) | <u>42.8</u> (49.7) | <b>2.7</b>  | <u>8.1</u>  |
| e5 (DiskANN)                          | 38.3 (68.5)        | 44.3 (44.3)        | 45.3 (45.3)        | 42.8 (42.8)        | 19.3 (24.2)        | 30.2 (35.5)        | 27.3 (35.9)        | 9.3 (12.1)         | 42.1 (49.0)        | <b>2.7</b>  | 8.0         |
| top-2                                 | 39.6 (70.7)        | 51.2 (51.2)        | 48.6 (48.6)        | 45.5 (45.5)        | 15.6 (28.0)        | <b>32.9</b> (40.6) | 25.7 (35.2)        | 7.6 (13.1)         | <b>43.1</b> (49.3) | <b>2.7</b>  | <b>8.3</b>  |
| top-10                                | 40.4 (70.7)        | 51.2 (51.2)        | 48.6 (48.6)        | 45.5 (45.5)        | <u>20.5</u> (29.9) | <u>33.2</u> (39.8) | 27.1 (36.1)        | <u>9.9</u> (14.3)  | 36.1 (48.9)        | <b>2.7</b>  | <u>8.1</u>  |

Table 2: Downstream performance on KILT dev set. Following Petroni et al. (2021), we report the results of typical metrics for each dataset, with bold indicating the best result and underlined indicating the second. The metrics with the prefix *KILT*- award output performance only when R-Prec = 1 (retrieval success). The figures in parentheses represent has\_answer percentage, which corresponds to the proportion of questions with gold answers included in the final output. The figures shown in gray are copied from the column above because they do not change based on the given setting (we use the Identity function of RALLE for the tasks, rather than an LLM). <sup>◇</sup>: Results from Petroni et al. (2021).

Our constructed R-LLM (e5 + Llama2-70B) surpasses the performance of the RAG model on both HoPo and TQA, despite not being fine-tuned with KILT like RAG. Moreover, our constructed R-LLMs demonstrate acceptable accuracy levels on other datasets as well, without any significant drawbacks. The results indicate that the LLMs used in this study exhibit certain ability to comprehend the retrieved documents.

Furthermore, our analysis reveals several factors that could contribute to improvement of downstream performance, including retrieval augmentation (except ELI5), increased model scale (except FEV and T-REx), and referring to more documents during generation (except NQ, HoPo, TQA and WoW). However, some datasets exhibits exceptions to these tendencies or had lower performance compared to their corresponding has\_answer percentage (such as FEV, T-REx, NQ, and TQA). To address this issue, developers can improve the R-LLM with RALLE by refining the inference chain and the prompt templates. In Section A.4, we provide our initial attempts at developing inference

chains with three actions on several datasets.

Overall, the downstream evaluation results provide valuable insights into how well the constructed R-LLMs perform on knowledge-intensive tasks, enabling developers to identify areas for improvement.

### 4.3 Retrieval Performance

Table 3 shows retrieval performance of the chosen retrievers on KILT development set (see also Table 8 in Appendix for the results of recall@5). According to Table 3, e5 (with Faiss Flat index) achieves the highest retrieval performance on average, though m-e5 is better on MTEB Retrieval task (Table 1). Despite the superior retrieval accuracy of e5 compared to RAG on KILT, the downstream performance of the R-LLM which employs e5 falls short of that of RAG (Table 2). This indicates that there is potential room for improvement through further optimized prompts to enhance the performance on a target dataset.

As described in Section 3.3, REWRITE-EL serves as the default template for search queries

| Dataset          | Fact Check. | Entity Linking |             |              | Slot Filling |             | Open Domain QA |             |             |             | Dial.       | Avg.         |
|------------------|-------------|----------------|-------------|--------------|--------------|-------------|----------------|-------------|-------------|-------------|-------------|--------------|
|                  | FEV         | AY2            | WnWi        | WnCw         | T-REx        | zsRE        | NQ             | HoPo        | TQA         | ELI5        | WoW         |              |
| Model            | R-Precision |                |             |              |              |             |                |             |             |             |             |              |
| RAG <sup>◇</sup> | 63.5        | <b>77.4</b>    | 49.0        | 46.7         | 29.3         | 65.4        | 60.3           | 30.8        | 49.3        | <b>16.4</b> | 46.7        | 48.6         |
| BM25             | 52.1        | 17.7           | 20.6        | 15.3         | 34.0         | 57.7        | 26.3           | 41.3        | 31.7        | 6.8         | 28.8        | 30.2         |
| – REWRITE-EL     | 52.1        | 3.0 (−14.7)    | 0.1 (−20.5) | 2.8* (−12.5) | 34.0         | 57.7        | 26.3           | 41.3        | 31.7        | 6.8         | 28.8        | 25.9 (−4.3)  |
| m-e5 (Flat)      | 81.7        | 41.8           | 45.8        | 41.6         | <b>47.1</b>  | 81.4        | 63.0           | 54.0        | <b>56.1</b> | 11.9        | <b>57.9</b> | 52.9         |
| – REWRITE-EL     | 81.7        | 3.2 (−38.6)    | 0.1 (−45.7) | 3.1 (−38.5)  | 47.1         | 81.4        | 63.0           | 54.0        | 56.1        | 11.9        | 57.9        | 41.8 (−11.1) |
| e5 (Flat)        | <b>82.0</b> | 51.6           | <b>51.6</b> | <b>49.2</b>  | 45.3         | <b>81.9</b> | <b>65.2</b>    | <b>54.3</b> | <b>56.1</b> | 12.9        | 56.8        | <b>55.2</b>  |
| – REWRITE-EL     | 82.0        | 3.4 (−48.2)    | 0.0 (−51.6) | 2.6 (−46.6)  | 45.3         | 81.9        | 65.2           | 54.3        | 56.1        | 12.9        | 56.8        | 41.9 (−13.3) |
| e5 (HNSW)        | 67.9        | 38.9           | 42.3        | 40.5         | 23.1         | 53.0        | 60.3           | 34.9        | 50.4        | 10.2        | 54.5        | 43.3         |
| – REWRITE-EL     | 67.9        | 2.9 (−36.0)    | 0.0 (−42.3) | 1.6 (−38.9)  | 23.1         | 53.0        | 60.3           | 34.9        | 50.4        | 10.2        | 54.5        | 32.6 (−10.7) |
| e5 (DiskANN)     | 78.8        | 44.7           | 47.8        | 46.0         | 37.1         | 74.5        | 64.9           | 49.1        | 55.4        | 12.9        | 56.6        | 51.6         |
| – REWRITE-EL     | 78.8        | 3.2 (−41.5)    | 0.1 (−47.7) | 1.8 (−44.2)  | 37.1         | 74.5        | 64.9           | 49.1        | 55.4        | 12.9        | 56.6        | 39.4 (−12.2) |

Table 3: Retrieval performances on KILT dev set. We report page-level R-Precision on KILT development set. Avg. refers to macro-average of the retrieval scores in each dataset. Bold indicates the best result. <sup>◇</sup>: Results from Petroni et al. (2021). \*: BM25 (without REWRITE-EL) failed with long queries (45 out of 5,599 questions) in WnCw.

| Retrieval                             |             |         |       |
|---------------------------------------|-------------|---------|-------|
| Model                                 | Avg. R-Prec | Memory  | sec/Q |
| BM25                                  | 30.2        | -       | 0.121 |
| e5 (Flat)                             | 55.2        | 84.8 GB | 0.169 |
| e5 (HNSW)                             | 43.3        | 90.4 GB | 0.008 |
| e5 (DiskANN)                          | 51.6        | 10.9 GB | 0.022 |
| Completion in the Closed-Book Setting |             |         | sec/Q |
| Llama-70B                             |             |         | 6.727 |
| Retrieval + Generation                |             |         | sec/Q |
| BM25 + Llama2-70B                     |             |         | 3.637 |
| e5 + Llama2-70B                       |             |         | 3.793 |
| e5 (DiskANN) + Llama2-70B             |             |         | 3.628 |

Table 4: Execution latency in seconds per question (sec/Q). Memory in Retrieval indicates the maximum (DRAM) memory footprints.

related to entity linking task (AY2, WnWi, and WnCw). As shown in Table 3, employing the REWRITE-EL template leads to higher retrieval accuracy when compared to using the full question text as a search query (– REWRITE-EL setting). This indicates that omitting unnecessary information from the search queries is helpful especially for entity linking task.

#### 4.4 Speed Analysis

RALLE allows users to optimize the trade-off between latency (in seconds per question) and accuracy by comparing various configurations. As demonstrated in Table 4, employing approximate nearest neighbor search (ANNS) algorithms such as HNSW and DiskANN can significantly reduce

retrieval latency at the cost of decreased accuracy. Note that, the optimal balance between speed and accuracy depends on the specific requirements of the application, and RALLE enables users to easily experiment with diverse ANNS settings to determine their impact on both factors.

Notably, DiskANN achieves an accuracy that is only slightly lower than Faiss flat index while significantly improving search speeds, despite requiring less memory footprints than both flat and HNSW indices. Though the reduction in R-LLM execution time achieved through ANNS may appear relatively minor, the significantly lower DRAM requirements of DiskANN could make it a more practical solution for scenarios where DRAM capacity is limited and the flat index exceeds available DRAM capacity. For further details regarding latency, refer to Table 9 in Appendix A.8.

## 5 Conclusion

This paper introduces RALLE, an accessible framework for developing and evaluating R-LLMs. We also report evaluation results of several R-LLMs built using open-source retrievers and LLMs on knowledge-intensive tasks. Overall, RALLE offers a significant advancement in retrieval-augmented generation research, enabling efficient development, evaluation, and improvement of R-LLMs. We hope that RALLE will contribute to the development of best practices for R-LLMs.

## Limitations

All KILT evaluations presented in this paper were conducted using a development set to maintain fair-

ness and consistency across evaluations, as the answers of the test set remain confidential<sup>11</sup>.

While R-LLMs exhibit high validity, it falls behind the smaller yet specialized model, RAG, on the KILT downstream task (refer to Table 2). This disparity can be attributed to various factors, including prompt maturity and the ability of LLMs to generate responses. Although the employed prompts were carefully developed, it is likely that more optimal prompts exist (discussed in Section 4.3). Moreover, fine-tuning LLMs with retrieval-augmented generation tasks might enhance their performance on downstream tasks. Therefore, the evaluation accuracy reported herein would represent a conservative estimate.

Prompt engineering is a crucial aspect of the retrieval-augmented generation process, as the generated outputs can differ significantly between models, even when provided with the same prompt. RALLE offers an advantage in this regard, allowing users to effortlessly experiment with diverse prompts for varying behaviors, datasets, and intricate chain of actions.

In the realm of prompt development, techniques like Automatic Prompt Engineer (APE) (Zhou et al., 2023) automate the creation of prompts from input-output pairs and sampling to identify the most effective prompts. However, the input-output pairs in retrieval-augmented generation are distinctly different from those of the simple instruction induction tasks. Because the input text for retrieval-augmented generation can often be lengthy and complex, it is difficult to automatically induce the effective prompts from the input-output pairs.

This tool enables developers to construct an inference chain with predefined actions, while recent advances have also introduced methods allowing LLMs to determine the actions (Yao et al., 2023). One approach entails retrieving documents using a query rewritten by an LLM and then summarizing them until the desired information is obtained. However, in our initial experiments (not described in this paper), we observed instances where relatively small LLMs (typically less than 100 billion parameters) became trapped in cycles of repeated retrieval and summarization, hindering their ability to reach the final answer generation. Our tool addresses this issue by intentionally building explicit inference chains to avoid unintended operations.

<sup>11</sup><https://eval.ai/web/challenges/challenge-page/689/overview>

## 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*.
- Akari Asai and Eunsol Choi. 2021. [Challenges in information-seeking QA: Unanswerable questions and paragraph retrieval.](#) 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 1492–1504, Online. Association for Computational Linguistics.
- Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. [A multi-task, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity.](#) *arXiv preprint arXiv:2302.04023*.
- Ali Borji. 2023. [A categorical archive of ChatGPT failures.](#) *arXiv preprint arXiv:2302.03494*.
- 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. 2023. [LangChain.](https://langchain.com/) <https://langchain.com/>.
- 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.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing GPT-4 with 90%\\* ChatGPT quality.](#)
- 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 Pillai, 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](#). *arxiv:2204.02311*.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martić, Shane Legg, and Dario Amodei. 2017. [Deep reinforcement learning from human preferences](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Nick Craswell. 2016. *R-Precision*, pages 1–1. Springer New York, New York, NY.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. [Retrieval augmented language model pre-training](#). In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.
- Benjamin Heinzerling and Kentaro Inui. 2021. [Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, Online. Association for Computational Linguistics.
- Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. [Diskann: Fast accurate billion-point nearest neighbor search on a single node](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. [Billion-scale similarity search with GPUs](#). *IEEE Transactions on Big Data*, 7(3):535–547.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- 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.
- June Lee. 2023. [WizardVicunaLM](https://github.com/melodysdreamj/WizardVicunaLM). <https://github.com/melodysdreamj/WizardVicunaLM>.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. [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, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- LF Projects. 2023. [MLflow – a platform for the machine learning lifecycle](https://mlflow.org/). <https://mlflow.org/>.
- 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.
- Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien De Masson D’Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsonan-Mcmahon, Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. 2022. [StreamingQA: A benchmark for adaptation to new knowledge over time in question answering models](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13604–13622. PMLR.
- Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. [Query rewriting for retrieval-augmented large language models](#). *arXiv preprint arXiv:2305.14283*.
- Yu A. Malkov and D. A. Yashunin. 2020. [Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs](#). *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836.
- Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu,

- Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. [Augmented language models: a survey](#). *arXiv preprint arXiv:2302.07842*.
- Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. [MTEB: Massive text embedding benchmark](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. [Webgpt: Browser-assisted question-answering with human feedback](#). *arXiv preprint arXiv:2112.09332*.
- Youyang Ng, Daisuke Miyashita, Yasuto Hoshi, Yasuhiro Morioka, Osamu Torii, Tomoya Kodama, and Jun Deguchi. 2023. [SimplyRetrieve: A private and lightweight retrieval-centric generative ai tool](#). *arXiv preprint arXiv:2308.03983*.
- OpenAI. 2023. [GPT-4 technical report](#). *arXiv preprint arXiv:2303.08774*, abs/2303.08774.
- 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.
- Ori Ram, Liat Bezael, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2022. [What are you token about? dense retrieval as distributions over the vocabulary](#). *arXiv preprint arXiv:2212.10380*.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: BM25 and beyond](#). *Found. Trends Inf. Retr.*, 3(4):333–389.
- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. [RePlug: Retrieval-augmented black-box language models](#). *arXiv preprint arXiv:2301.12652*.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. [Learning to summarize with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.
- 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. 2023a. [LLaMA: Open and efficient foundation language models](#). *arXiv preprint arXiv:2302.13971*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. [Llama 2: Open foundation and fine-tuned chat models](#). *arXiv preprint arXiv:2307.09288*.
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. [Text embeddings by weakly-supervised contrastive pre-training](#). *arXiv preprint arXiv:2212.03533*.
- 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.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. [WizardLM: Empowering large language models to follow complex instructions](#). *arXiv preprint arXiv:2304.12244*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwon Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. [Large language models are human-level prompt engineers](#). In *The Eleventh International Conference on Learning Representations*.

## A Appendix

### A.1 Computational Resources

The evaluation experiments are conducted on an Ubuntu 20.04.6 server equipped with Intel(R) Xeon(R) Gold 6326 CPU at 2.90 GHz CPU cores, and one node with 4×NVIDIA A100 Tensor Core GPU with 40 GB memory, and a RAID-5 array with a Dell(R) PERC H745 Front controller and KIOXIA(R) PM6-R SAS SSDs for storage. The CUDA version is 12.2, the Python version is 3.9.16, the PyTorch version is 2.0.1, and the Transformers version is 4.29.2.



Load models   Develop chain   Chat   Config

Dataset: NQ

Question: who had the most wins in the nfl

Config: [dropdown]   Load config   Chain length: 2   Execute entire chain

---

**Action 1**

**Interpret prompt only**   **Interpret prompt and execute this action**

prompt\_template[0]: {question}

prompt[0]: who had the most wins in the nfl

response[0]: following is list of top national football league nfl quarterbacks in wins in nfl quarterback is only position that is credited with records of wins and losses active quarterback **tom brady** holds records for most wins with 237 most regular season wins wi

Function:  LLM    Retriever    Identity

Retriever name: e5-8b\_flat   k, press ENTER key to apply: 5

wiki\_id\_title[0]: List of National Football League career quarterback wins leaders, **13929036**; Super Bowl, 27718; List of Super Bowl champions, 2145410; List of National Football League head coaches with 50 wins, 41018355; List of National Football League career quarterback wins leaders, **13929036**

---

**Action 2**

**Interpret prompt only**   **Interpret prompt and execute this action**

prompt\_template[1]: Referring to the following document, answer "{question}" in 5 words or less.

prompt[1]: Referring to the following document, answer "who had the most wins in the nfl?" in 5 words or less.

response[1]: **tom brady**

Function:  LLM    Retriever    Identity

LLM name: llama-2-70b-chat

plain or highlighted:  plain text    highlighted text

---

show qa data

```

{
  qa_data: [
    {
      id: "5289242154789678439",
      input: "who had the most wins in the nfl",
      output: [
        {
          answer: "Tom Brady",
          provenance: [
            {
              wikipedia_id: "13929036",
              title: "List of National Football League career quarterback wins leaders",
              start_paragraph_id: 2,
              start_character: 19,
              end_paragraph_id: 2,
              end_character: 28,
              bleu_score: 1,
              section: "Section:::Abstract."
            }
          ]
        }
      ]
    }
  ]
}

```

Figure 2: A screenshot of the *Development chain* tab of RALLE. Developers can create tailored action chains comprising multiple actions of inference. For each action, developers can specify a prompt template, confirm the results of applying the template, and execute the action using the newly defined prompt, individually. Moreover, RALLE can highlight the gold answers within the retrieved documents or the output of the LLM, as well as highlight the Wikipedia IDs of successfully retrieved provenance.

## A.2 Development Screen of RALLE

Figure 2 shows the chain development screen<sup>12</sup>. Developers can create an inference chain for an R-LLM on this *Develop chain* tab. One can choose a dataset and specify the desired chain length, which represents the total number of actions. By default, there are two actions: retrieving with a retriever and generating with an LLM.

Prompt templates for each action can be defined using f-strings or eval functions in Python. The results of applying the template can be confirmed without executing retrieval and generation. The execution result can be viewed by clicking the *Interpret prompt and execute this action* button.

The available action operators are *LLM*, *Retriever*, and *Identity*. *LLM* generates text based on the given prompt. *Retriever* retrieves the top  $k$  most relevant documents related to the input query. And *Identity* simply outputs the original prompt without employing a retriever or an LLM.

To execute the entire chain, click the *Execute entire chain* button. At the bottom of this tab, the selected question and its corresponding answer can be reviewed. Also, RALLE enables to highlight the gold answers within the retrieved documents or the output of the LLM, as well as highlight the Wikipedia ID of successfully retrieved provenance.

## A.3 Additional Metric: has\_answer

RALLE also includes `has_answer` percentage (e.g., Karpukhin et al., 2020) for short answers, a proxy metric to measure the proportion of questions that contain gold answers within the final output generated by an R-LLM. By tracking this metric, developers can identify situations where the model generates responses that include gold answers but may be overlooked due to evaluation biases such as exact matching. This information can help refine prompts to improve overall performance.

## A.4 Attempts to Build 3-action Chain

According to Section 4.2, retrieval augmentation has a significant impact on performance in fact checking, open-domain QA for short answers, and slot filling tasks when comparing the closed-book and open-book settings of Llama2-70B. In entity linking task (AY2, WnWi, and WnCw), however, our approach described in Section 3.3 (retrieve, then output the top-1 retrieved Wikipedia title) may not be effective.

<sup>12</sup>Please also review the [demonstration screencast](#).

To improve the performance, we construct a 3-action chain for AY2 dataset: (1) retrieve top-5 relevant documents, (2) explain the entity mention being questioned, and (3) predict the Wikipedia title based on the explanation and top-5 retrieved titles. Additionally, we explore developing 3-action chains for T-REx and NQ datasets, which involves (1) retrieval, (2) question rewriting, and (3) answer generation. Table 12 shows the prompts used in 3-action chains.

Table 5 shows the downstream performances with the 3-action chains on AY2, NQ, and T-REx datasets. While the 3-action chain outperforms the 2-action (retrieve-then-generate) chain on NQ dataset, it underperforms the 2-action accuracies on AY2 and T-REx datasets. This suggests that the 3-action chains constructed specifically for these two datasets require further optimization. However, the `has_answer` value for AY2 (70.0%) is higher than that of the 2-action chain (47.8%), indicating that incorporating post-processing steps into the 3-action chain (thus to be 4-action chain) could potentially boost accuracy, particularly for AY2.

One of the benefits of our tool is that it allows for easy definition of such additional inference actions. This means that developers can customize the chain to perform specific tasks beyond the default setting, giving them greater flexibility and control over their development.

## A.5 Details of Baseline Model in Open-Book Setting

As a baseline in open-book setting, we present the results of the Retrieval-Augmented Generation (RAG) model (Lewis et al., 2020b) shown in Petroni et al. (2021), which achieved strong performance in the KILT benchmark. The RAG model comprises a bi-encoder retriever and a sequence-to-sequence generator (BART model (Lewis et al., 2020a)), both of which are trained end-to-end. The total number of trainable parameters in the RAG model is approximately 626 million. It is important to note that the RAG model was trained specifically for the KILT benchmark, whereas our chosen LLMs and constructed R-LLMs were not.

## A.6 KILT Downstream Performances in Closed-Book Setting

Table 6 summarizes the KILT downstream results in a closed-book setting. The baseline (BART-large) model has been fine-tuned on the KILT

| Dataset                           | Fact Check.   | Entity Linking |             |             | Slot Filling |             | Open Domain QA |                    |                    |             | Dial.       |
|-----------------------------------|---------------|----------------|-------------|-------------|--------------|-------------|----------------|--------------------|--------------------|-------------|-------------|
|                                   | FEV           | AY2            | WnWi        | WnCw        | T-REx        | zsRE        | NQ             | HoPo               | TQA                | ELI5        | WoW         |
| Model / Metric                    | Accuracy      |                |             |             |              |             | Exact Match    |                    |                    | RL          | F1          |
| Llama2-70B ( <i>closed-book</i> ) | 33.6 (74.9)   | 39.8 (54.5)    | 42.8 (53.8) | 39.2 (55.7) | 28.5 (40.5)  | 11.3 (13.6) | 19.6 (37.4)    | 13.9 (25.1)        | 67.4 (80.8)        | <b>23.0</b> | <b>13.3</b> |
| RAG <sup>◇</sup>                  | <b>87.7</b>   | <b>77.4</b>    | <b>49.0</b> | <b>46.7</b> | <b>61.5</b>  | <b>47.4</b> | <b>48.8</b>    | 27.7               | 61.7               | 16.1        | <b>13.3</b> |
| e5 + Llama2-70B                   | 49.9 (88.6)   | 51.2 (57.9)    | 48.6 (51.4) | 45.6 (51.4) | 28.9 (49.2)  | 35.0 (43.2) | 36.4 (48.8)    | <b>28.1</b> (35.8) | <b>71.1</b> (83.9) | 21.5        | 13.2        |
| 3-action                          | -             | 24.4 (70.0)    | -           | -           | 16.3 (46.8)  | -           | 36.9 (49.3)    | -                  | -                  | -           | -           |
| Model / Metric                    | KILT-Accuracy |                |             |             |              |             | KILT-EM        |                    | KILT-RL            | KILT-F1     |             |
| RAG <sup>◇</sup>                  | <b>55.5</b>   | <b>77.4</b>    | <b>49.0</b> | <b>46.7</b> | <b>25.4</b>  | <b>42.6</b> | <b>36.3</b>    | 3.1                | 36.1               | <b>2.7</b>  | 7.5         |
| e5 + Llama2-70B                   | 40.2 (71.2)   | 51.2 (51.2)    | 48.6 (48.6) | 45.5 (45.5) | 19.2 (29.7)  | 32.8 (40.4) | 27.7 (36.3)    | <b>11.3</b> (14.5) | <b>42.8</b> (49.7) | <b>2.7</b>  | <b>8.1</b>  |
| 3-action                          | -             | 9.5 (27.7)     | -           | -           | 10.4 (27.9)  | -           | 28.0 (36.6)    | -                  | -                  | -           | -           |

Table 5: Downstream performance of the 3-action chain on KILT dev set along with baselines. The figures in parentheses represent has\_answer percentage, which corresponds to the proportion of questions with gold answers included in the final output of the LLM. <sup>◇</sup>: Results from Petroni et al. (2021).

| Dataset                 | Fact Check. | Entity Linking |             |             | Slot Filling |                    | Open Domain QA |             |                    |             | Dial.       |
|-------------------------|-------------|----------------|-------------|-------------|--------------|--------------------|----------------|-------------|--------------------|-------------|-------------|
|                         | FEV         | AY2            | WnWi        | WnCw        | T-REx        | zsRE               | NQ             | HoPo        | TQA                | ELI5        | WoW         |
| Model / Metric          | Accuracy    |                |             |             |              |                    | Exact Match    |             |                    | RL          | F1          |
| BART-large <sup>◇</sup> | <b>80.7</b> | <b>86.6</b>    | <b>47.9</b> | <b>48.0</b> | <b>43.8</b>  | 3.0                | <b>26.2</b>    | <b>16.9</b> | 32.5               | 22.7        | <b>13.8</b> |
| W-Vicuna-13B            | 0.0 (58.4)  | 0.1 (52.2)     | 2.0 (44.9)  | 0.0 (48.1)  | 17.9 (33.0)  | 5.9 (8.5)          | 6.2 (27.4)     | 1.7 (17.1)  | 20.0 (64.5)        | 22.7        | 12.7        |
| Llama2-13B              | 26.3 (50.7) | 34.6 (47.5)    | 35.0 (42.8) | 28.5 (41.3) | 26.9 (36.7)  | 7.8 (9.9)          | 11.5 (29.1)    | 8.3 (20.3)  | 43.0 (70.2)        | <b>27.6</b> | 13.0        |
| Llama2-70B              | 33.6 (74.9) | 39.8 (54.5)    | 42.8 (53.8) | 39.2 (55.7) | 28.5 (40.5)  | <b>11.3</b> (13.6) | 19.6 (37.4)    | 13.9 (25.1) | <b>67.4</b> (80.8) | 23.0        | 13.3        |

Table 6: Downstream performance on KILT development set in a *closed-book* setting (generation without retrieval). Following Petroni et al. (2021), we report the results of typical metrics for each dataset, with bold indicating the best result. The figures in parentheses represent has\_answer percentage, which corresponds to the proportion of questions with gold answers included in the final output of the LLM. <sup>◇</sup>: Results from Petroni et al. (2021).

datasets, while our chosen LLMs have not. Despite this, the LLMs demonstrate superior performance compared to the baseline on several datasets.

Specifically, the Llama2-70B model outperforms the BART baseline on the zsRE and TQA datasets, and the Llama2-13B model outperforms the baseline on the ELI5 dataset. This suggests that the parametric knowledge embedded in the LLMs and their capacity for text generation can be leveraged effectively for knowledge-intensive tasks, even zero-shot setting. Nevertheless, as described in Section 4.2, retrieval augmentation can enhance the performance on downstream tasks, except the ELI5 dataset. We also present the closed-book performances of several LLMs on the development set of NQ dataset in Table 7.

### A.7 Additional Results for Retrieval Performance

Table 8 presents the recall@5 of the retrievers used in our experiments. Note that even though m-e5 outperforms e5 on the MTEB Retrieval task (shown in Table 1), e5 still demonstrates superior performance compared to m-e5 in terms of both

R-precision (shown in Table 3) and recall@5.

### A.8 Details of Speed Analysis

Table 9 presents the details of speed analysis on KILT development set. The search speed of BM25 (without REWRITE-EL) decreases as the total number of words in a query increases. In contrast, for dense vector search, the search speed remains relatively constant regardless of the size of the query due to the fixed dimensionality of the embedding vectors.

According to Table 9, the execution times required for generation with an LLM is longer than the times required for retrieval, particularly when generating lengthy responses such as ELI5 and WoW. Therefore, it may seem counterintuitive that the advantages of ANNS used in vector search are not fully realized in terms of execution time of R-LLMs. However, as previously discussed in Section 4.4, DiskANN requires less memory compared to other vector search algorithms, which means that using such algorithm can actually help conserve computational resources for R-LLM.

We observe that Llama2-13B requires more time

| NQ               |             |             |             |       |
|------------------|-------------|-------------|-------------|-------|
| Model Name       | EM          | has_answer  | f1          | sec/Q |
| Llama-2-70b-chat | 19.6        | 37.4        | 36.8        | 2.254 |
| Llama-2-13b-chat | 11.5        | 29.1        | 28.1        | 1.179 |
| StableBeluga2    | 16.2        | <b>40.9</b> | 35.5        | 2.858 |
| gpt-3.5-turbo    | <b>25.4</b> | 38.9        | <b>41.1</b> | -     |

Table 7: Accuracies on NQ dev set in a closed-book setting. For gpt-3.5-turbo (version 0613), the accuracy was calculated excluding five questions out of 2,837 questions in the NQ development set that were deemed inappropriate prompts by OpenAI and were not processed.

| Dataset                  | Fact Check. |             | Entity Linking |              |             | Slot Filling |             | Open Domain QA |             |             | Dial.       | Avg.         |
|--------------------------|-------------|-------------|----------------|--------------|-------------|--------------|-------------|----------------|-------------|-------------|-------------|--------------|
|                          | FEV         | AY2         | WnWi           | WnCw         | T-REx       | zsRE         | NQ          | HoPo           | TQA         | ELI5        | WoW         |              |
| Model                    | Recall@5    |             |                |              |             |              |             |                |             |             |             |              |
| RAG <sup>◇</sup>         | 76.1        | <b>77.5</b> | 49.0           | 46.7         | 33.7        | 73.1         | 65.5        | 12.3           | 56.9        | <b>27.3</b> | 66.6        | 53.1         |
| BM25                     | 74.2        | 28.8        | 34.7           | 30.6         | 42.7        | 74.7         | 42.5        | 22.8           | 48.7        | 12.3        | 45.1        | 41.6         |
| – REWRITE-EL             | 74.2        | 7.6 (–21.2) | 3.1 (–31.6)    | 5.9* (–24.7) | 42.7        | 74.7         | 42.5        | 22.8           | 48.7        | 12.3        | 45.1        | 34.5 (–7.1)  |
| m-e5 (Flat)              | <b>91.0</b> | 58.5        | 60.6           | 62.2         | <b>53.1</b> | 87.0         | 69.5        | 40.4           | <b>65.4</b> | 19.1        | 75.0        | 62.0         |
| – REWRITE-EL             | 91.0        | 7.8 (–50.7) | 3.8 (–56.8)    | 5.5 (–56.7)  | 53.1        | 87.0         | 69.5        | 40.4           | 65.4        | 19.1        | 75.0        | 47.1 (–14.9) |
| m-e5 (HNSW) – REWRITE-EL | 63.2        | 4.9         | 3.5            | 2.6          | 26.0        | 48.2         | 55.6        | 14.1           | 48.7        | 14.6        | 66.8        | 31.7         |
| e5 (Flat)                | 90.6        | 66.1        | <b>63.3</b>    | <b>66.7</b>  | 52.1        | <b>87.2</b>  | <b>71.6</b> | <b>40.9</b>    | <b>65.4</b> | 21.3        | <b>75.3</b> | <b>63.7</b>  |
| – REWRITE-EL             | 90.6        | 7.6 (–58.5) | 3.4 (–59.9)    | 4.8 (–61.9)  | 52.1        | 87.2         | 71.6        | 40.9           | 65.4        | 21.3        | 75.3        | 47.3 (–16.4) |
| e5 (HNSW)                | 74.7        | 49.6        | 50.7           | 50.8         | 26.7        | 55.9         | 65.2        | 19.3           | 58.6        | 16.0        | 70.9        | 48.9         |
| – REWRITE-EL             | 74.7        | 6.0 (–43.6) | 3.3 (–47.4)    | 3.3 (–47.5)  | 26.7        | 55.9         | 65.2        | 19.3           | 58.6        | 16.0        | 70.9        | 36.4 (–12.5) |
| e5 (DiskANN)             | 86.6        | 57.1        | 58.3           | 60.9         | 42.1        | 78.7         | 70.7        | 34.7           | 64.6        | 20.8        | 75.0        | 59.0         |
| – REWRITE-EL             | 86.6        | 7.4 (–49.7) | 3.3 (–55.0)    | 3.6 (–57.3)  | 42.1        | 78.7         | 70.7        | 34.7           | 64.6        | 20.8        | 75.0        | 44.3 (–14.7) |

Table 8: Retrieval performances (recall@5) on KILT dev set. Avg. refers to macro-average of the scores in each dataset. Bold indicates the best result. The figures shown in gray are copied from the column above because they do not change based on the given setting. <sup>◇</sup>: Results from Petroni et al. (2021). \*: BM25 (without REWRITE-EL) failed with long queries (45 out of 5,599 questions) in WnCw.

to process each question compared to Llama2-70B. Upon further analysis, we discovered that the Llama2-13B model occasionally produced nonsensical responses such as multiple newline characters (“\n”), partially due to the limitations of our prompts.

## A.9 Model Information

As shown in Table 10, we utilize several open-source models from Hugging Face, specifically their officially released versions. We load the distributed models in 8-bit precision by default except Llama2-70B model (in 4-bit) using Hugging Face Accelerate<sup>13</sup> library.

## A.10 Using Custom Datasets

In addition to utilizing KILT datasets, RALLE enables developers to develop and evaluate R-LLMs on their own QA datasets and corpora. To use the

<sup>13</sup><https://huggingface.co/docs/accelerate/index>

custom datasets with RALLE, you will need to perform the following preprocessing:

- Prepare your corpus as a TSV file containing the document IDs, texts, and titles.
- Create a JSONL file for your QA dataset. The format should look like this: {"id": "", "input": "", "output": [{"answer": "", "provenance": [{"wikipedia\_id": "", "title": ""}]}]}, where “input” represents a question.

See our repo for more detailed instructions: <https://github.com/yhoshi3/RaLLe>.

## A.11 The Prompts used in the Evaluation

Table 11 summarizes the prompts used in our experiment. *Open-book* indicates retrieve-then-generate setting. The queries used for retrieval are the raw questions without any rewriting, except for the REWRITE-EL settings of AY2, WnWi, and WnCw.

| Tasks                      | Fact Check.   | Entity Linking |             |            | Slot Filling |         | Open Domain QA |          |          |           | Dial.     | Avg.  |
|----------------------------|---|----------------|-------------|------------|--------------|---------|----------------|----------|----------|-----------|-----------|-------|
|                            | FEV   | AY2            | WnWi        | WnCw       | T-REx        | zsRE    | NQ             | HoPo     | TQA      | ELIS      | WoW       |       |
| <i>Models</i>              | Completion in Closed-Book Setting (in seconds per question) |                |             |            |              |         |                |          |          |           |           |       |
| W-Vicuna-13B               | 1.565   | 13.040         | 10.870      | 9.793      | 0.983        | 1.142   | 2.165          | 1.969    | 1.414    | 22.820    | 7.122     | 6.626 |
| Llama2-13B                 | 0.625   | 1.077          | 1.036       | 1.201      | 0.940        | 0.913   | 1.270          | 1.185    | 1.014    | 40.100    | 9.522     | 5.353 |
| Llama2-70B                 | 1.765   | 2.936          | 2.745       | 2.618      | 1.953        | 2.031   | 2.285          | 2.188    | 1.877    | 42.500    | 11.100    | 6.727 |
|                            | Retrieval + Generation (in seconds per question)            |                |             |            |              |         |                |          |          |           |           |       |
| e5 + W-Vicuna-13B          | 1.529   | 1.310          | 1.368       | 1.158      | 1.192        | 1.453   | 2.595          | 1.945    | 1.734    | 15.480    | 10.850    | 3.692 |
| e5 + Llama2-13B            | 1.084   | 1.165          | 1.209       | 1.046      | 1.300        | 1.407   | 1.284          | 1.975    | 9.830    | 32.48     | 16.76     | 6.322 |
| BM25 + Llama2-70B          | 1.841   | 0.008          | 0.009       | 0.008      | 2.015        | 2.296   | 2.206          | 2.344    | 2.249    | 15.020    | 12.010    | 3.637 |
| e5 + Llama2-70B            | 1.926   | 0.133          | 0.131       | 0.135      | 2.135        | 2.424   | 2.419          | 2.346    | 2.238    | 16.030    | 11.810    | 3.793 |
| e5 (top-2) + Llama2-70B    | 1.544   | 0.133          | 0.131       | 0.135      | 1.661        | 1.908   | 1.994          | 1.833    | 1.759    | 15.120    | 10.820    | 3.367 |
| e5 (top-10) + Llama2-70B   | 2.811   | 0.133          | 0.131       | 0.135      | 2.951        | 3.276   | -              | 13.900   | 14.400   | 35.070    | 24.100    | -     |
| e5 (DiskANN) + Llama2-70B  | 1.803   | 0.044          | 0.044       | 0.043      | 2.009        | 2.281   | 2.166          | 2.247    | 2.116    | 15.780    | 11.370    | 3.628 |
| e5 + Llama2-70B (3-action) | -   | 25.41          | -           | -          | 4.993        | -       | 16.320         | -        | -        | -         | -         | -     |
|                            | Retrieval (in seconds per question)                         |                |             |            |              |         |                |          |          |           |           |       |
| BM25                       | 0.038   | 0.008          | 0.009       | 0.008      | 0.018        | 0.013   | 0.052          | 0.105    | 0.086    | 0.136     | 0.857     | 0.121 |
| BM25 (without REWRITE-EL)  | 0.038   | 5.700          | 4.531       | 5.440      | 0.018        | 0.013   | 0.052          | 0.105    | 0.086    | 0.136     | 0.857     | 1.543 |
| m-e5 (Flat)                | 0.174   | 0.164          | 0.166       | 0.176      | 0.187        | 0.165   | 0.194          | 0.156    | 0.176    | 0.177     | 0.165     | 0.173 |
| m-e5 (HNSW)                | 0.008   | 0.013          | 0.013       | 0.015      | 0.008        | 0.009   | 0.009          | 0.009    | 0.009    | 0.011     | 0.010     | 0.010 |
| e5 (Flat)                  | 0.177   | 0.168          | 0.172       | 0.159      | 0.201        | 0.170   | 0.171          | 0.146    | 0.155    | 0.174     | 0.165     | 0.169 |
| e5 (HNSW)                  | 0.008   | 0.008          | 0.008       | 0.008      | 0.008        | 0.008   | 0.008          | 0.008    | 0.008    | 0.008     | 0.009     | 0.008 |
| e5 (DiskANN)               | 0.018   | 0.020          | 0.038       | 0.020      | 0.020        | 0.030   | 0.020          | 0.021    | 0.019    | 0.019     | 0.021     | 0.022 |
|                            | Mean Query Length (tokens)                                  |                |             |            |              |         |                |          |          |           |           |       |
|                            | 11.1±4.0  | 357.9±149.0    | 331.5±113.5 | 505.2±31.1 | 7.5±2.5      | 7.6±2.3 | 9.9±2.1        | 19.5±6.6 | 17.9±8.9 | 21.0±10.7 | 86.3±58.0 |       |

Table 9: Execution time (in seconds per question) in RALLE. Avg. refers to macro-average of the times in each task. The mean query length and its standard deviation (shown as  $\pm$  after the value) are also displayed, which were calculated using the e5 tokenizer.

| Language Model                           |                |          |         |   |
|--|----------------|----------|---------|---|
| Model Name                               | Size           | max len. | emb dim | URL   |
| wizard-vicuna-13b (Lee, 2023)            | 13,015,864,320 | 2,048    | -       | <a href="https://huggingface.co/junelee/wizard-vicuna-13b">https://huggingface.co/junelee/wizard-vicuna-13b</a>           |
| Llama-2-13b-chat (Touvron et al., 2023b) | 13,015,864,320 | 4,096    | -       | <a href="https://huggingface.co/meta-llama/Llama-2-13b-chat">https://huggingface.co/meta-llama/Llama-2-13b-chat</a>       |
| Llama-2-70b-chat (Touvron et al., 2023b) | 68,976,653,312 | 4,096    | -       | <a href="https://huggingface.co/meta-llama/Llama-2-70b-chat">https://huggingface.co/meta-llama/Llama-2-70b-chat</a>       |
| StableBeluga2                            | 70B            | 4,096    | -       | <a href="https://huggingface.co/stabilityai/StableBeluga2">https://huggingface.co/stabilityai/StableBeluga2</a>           |
| Retriever                                |                |          |         |   |
| multilingual-e5-large                    | 559,890,946    | 514      | 1,024   | <a href="https://huggingface.co/intfloat/multilingual-e5-large">https://huggingface.co/intfloat/multilingual-e5-large</a> |
| e5-large-v2 (Wang et al., 2022)          | 335,142,400    | 512      | 1,024   | <a href="https://huggingface.co/intfloat/e5-large-v2">https://huggingface.co/intfloat/e5-large-v2</a>                     |

Table 10: Hugging Face links of the models used in our evaluation. Size refers to the total number of effective parameters of each model. max len. refers to the maximum token length of model input.

*Closed-book* indicates that an LLM answers to the given question without retrieval. Although these prompts have been our established best practices, we recognize that there may be opportunities for improvement (see also Section 5).

| <i>Open-book</i>  | <i>Closed-book</i>   |
|---|--|
| <b>FEVER</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval  |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>{response[0]} ← ← Answer IN ONE WORD if the document SUPPORTS or REFUTES "{question}". ← ← Answer:</pre>   | <p><b>Action 1: LLM</b></p> <pre>Answer IN ONE WORD if your knowledge SUPPORTS or REFUTES "{question}". ← ← Answer:</pre>  |
| <b>AY2</b> <input type="radio"/> f-strings <input checked="" type="radio"/> eval  |  |
| <p><b>Action 1: Retriever</b></p> <pre>'What is "' + '{'}.format(question).split( '[START_ENT']')[1].split('[END_ENT]')[0][1:-1] + "' ?'</pre> <p><b>Action 2: Identity</b></p> <pre>'{'}'.format(wiki_id_title[0]).split('; ')[0].split( ')[0]</pre> | <p><b>Action 1: LLM</b></p> <pre>'What is the most relevant Wikipedia title to the en- tity "' + '{'}.format(question).split('[START_ENT] ')[1].split('[END_ENT]')[0] + "' in the context of "' + '{'}.format(question).split('[START_ENT]')[0][-100:] + '{'}.format(question).split('[START_ENT]')[1].split( '[END_ENT]')[0] + '{'}.format(question).split( '[END_ENT]')[1][:100] + '...'?"\n\nPlease answer only the Wikipedia title.\n\nAnswer: '''</pre> |
| <b>WnWi</b> <input type="radio"/> f-strings <input checked="" type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>'What is "' + '{'}.format(question).split( '[START_ENT']')[1].split('[END_ENT]')[0][1:-1] + "' ?'</pre> <p><b>Action 2: Identity</b></p> <pre>'{'}'.format(wiki_id_title[0]).split('; ')[0].split( ')[0]</pre> | <p><b>Action 1: LLM</b></p> <pre>'What is the most relevant Wikipedia title to the en- tity "' + '{'}.format(question).split('[START_ENT] ')[1].split('[END_ENT]')[0] + "' in the context of "' + '{'}.format(question).split('[START_ENT]')[0][-100:] + '{'}.format(question).split('[START_ENT]')[1].split( '[END_ENT]')[0] + '{'}.format(question).split( '[END_ENT]')[1][:100] + '...'?"\n\nPlease answer only the Wikipedia title.\n\nAnswer: '''</pre> |
| <b>WnCw</b> <input type="radio"/> f-strings <input checked="" type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>'What is "' + '{'}.format(question).split( '[START_ENT']')[1].split('[END_ENT]')[0][1:-1] + "' ?'</pre> <p><b>Action 2: Identity</b></p> <pre>'{'}'.format(wiki_id_title[0]).split('; ')[0].split( ')[0]</pre> | <p><b>Action 1: LLM</b></p> <pre>'What is the most relevant Wikipedia title to the en- tity "' + '{'}.format(question).split('[START_ENT] ')[1].split('[END_ENT]')[0] + "' in the context of "' + '{'}.format(question).split('[START_ENT]')[0][-100:] + '{'}.format(question).split('[START_ENT]')[1].split( '[END_ENT]')[0] + '{'}.format(question).split( '[END_ENT]')[1][:100] + '...'?"\n\nPlease answer only the Wikipedia title.\n\nAnswer: '''</pre> |

*Continued on next page...*

Table 11: Prompt templates used in our experiments. The hook-left arrows ← refers to new line. Note that RALLE supports f-strings and eval() function in Python.

Table 11 – continued from previous page.

| <i>Open-book</i>  | <i>Closed-book</i>   |
|---|--|
| <b>T-REx</b>  |  |
| <p><b>Action 1: Retriever (f-strings)</b></p> <pre>{question}</pre> <p><b>Action 2: LLM (eval ())</b></p> <pre>'''Referring to the following document, answer "what is the ''' + '{}'.format(question).split('[SEP]')[1] + 'of ' + '{}'.format(question).split('[SEP]')[0] + '''?'' in 5 words or less.\n\n''' + '{}'.format(response[0]) + '''\n\n''' + '{}'.format(question).split('[SEP]')[1] + ''':</pre> | <p><b>Action 1: LLM (eval ())</b></p> <pre>'What is the ' + ''' + '{}'.for- mat(question).split('[SEP] ') [1] + ' of ' + '{}'.for- mat(question).split('[SEP]')[0] + ''' + ''' in 5 words or less?\n\n''' + '{}'.format(question).split('[SEP] ') [1] + ': '</pre> |
| <b>zsRE</b>   |  |
| <p><b>Action 1: Retriever (f-strings)</b></p> <pre>{question}</pre> <p><b>Action 2: LLM (eval ())</b></p> <pre>Referring to the following document, answer "{ques- tion}?" in 5 words or less. ← ← {response[0]} ← ← Answer:</pre>  | <p><b>Action 1: LLM (eval ())</b></p> <pre>'Tell me the ' + ''' + '{}'.for- mat(question).split('[SEP] ') [1] + ' of ' + '{}'.for- mat(question).split('[SEP]')[0] + ''' + ''' in 5 words or less.\n\n''' + '{}'.format(question).split('[SEP] ') [1] + ': '</pre> |
| <b>NQ</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>Referring to the following document, answer "{ques- tion}?" in 5 words or less. ← ← {response[0]} ← ← Answer:</pre>  | <p><b>Action 1: LLM</b></p> <pre>Answer '{question}?' in 5 words or less. ← ← Answer:</pre>  |
| <b>HoPo</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>Referring to the following document, answer "{ques- tion}?" in 5 words or less. ← ← {response[0]} ← ← Answer:</pre>  | <p><b>Action 1: LLM</b></p> <pre>Answer '{question}?' in 5 words or less. ← ← Answer:</pre>  |

Continued on next page...

Table 11 – continued from previous page.

| <i>Open-book</i>   | <i>Closed-book</i>   |
|--|--|
| <b>TQA</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>Referring to the following document, answer "{ques- tion}?" in 5 words or less. {response[0]} Answer:</pre>   | <p><b>Action 1: LLM</b></p> <pre>Answer '{question}' in 5 words or less. Answer:</pre>   |
| <b>ELI5</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval  |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>Referring to the following document, answer "{question}". {response[0]} Explain the following questions as if I were five years old. {question} Answer:</pre>   | <p><b>Action 1: LLM</b></p> <pre>Explain '{question}' as if I were five years old. Answer:</pre>   |
| <b>WoW</b> <input checked="" type="radio"/> f-strings <input type="radio"/> eval   |  |
| <p><b>Action 1: Retriever</b></p> <pre>{question}</pre> <p><b>Action 2: LLM</b></p> <pre>Referring to the following document, output a short and informative reply to the conversation. {response[0]} Referring to the above document, output a short and in- formative reply to the following conversation. This conversation ends on your turn. {question} Informative and short answer:</pre> | <p><b>Action 1: LLM</b></p> <pre>Output a short and informative reply to the conversation. This conversation ends on your turn. {question} Informative and short answer:</pre> |



---

AY2

f-strings

eval

---

**Action 1: Retriever**

```
'What is "' + '{}'.format(question).split(' [START_ENT] ')[1].split(' [END_ENT]')[0] + '" in the context of "' +
'{}'.format(question).split(' [START_ENT]')[0][-100:] + '{}'.format(question).split(' [START_ENT]')[1].split(' [END_ENT]')[0]
+ '{}'.format(question).split(' [END_ENT]')[1][:100] + '...?'
```

**Action 2: LLM**

```
'What is "' + '{}'.format(question).split(' [START_ENT] ')[1].split(' [END_ENT]')[0] + '" in the context of "' +
'{}'.format(question).split(' [START_ENT]')[0][-100:] + '{}'.format(question).split(' [START_ENT]')[1].split(' [END_ENT]')[0]
+ '{}'.format(question).split(' [END_ENT]')[1][:100] + '...'?
Answer in a short and conc sentence.' + '\n\nAnswer:\n''
```

**Action 3: LLM**

```
'Please select the most appropriate title for the word "' + '{}'.format(question).split(' [START_ENT]
')[1].split(' [END_ENT]')[0] + '" based on the given Description.' + '\n\nIf none of these titles suit your needs, please
suggest a possible alternative title.' + '\n\nTitles: \n'' + '/ '.join([titleid.split(',')[0] for titleid in '{}'.for-
mat(wiki_id_title[0]).split('; ')]) + '\n\nDescription:\n'' + '{}'.format(response[1]) + '\n\nWikipedia Title:\n''
```

---

T-REx

f-strings

eval

---

**Action 1: Retriever**

```
{question}
```

**Action 2: LLM**

```
Formulate a question that asks [SEP] in the following sentence: ←
'{question}' ←
←
Generated question:
```

**Action 3: LLM**

```
{response[0]} ←
←
Referring to the document above, answer "{response[1]}" in 5 words or less. ←
←
Answer:
```

---

NQ

f-strings

eval

---

**Action 1: Retriever**

```
{question}
```

**Action 2: LLM**

```
Please rewrite the following question clearly. ←
←
{question}? ←
←
Rewritten question:
```

**Action 3: LLM**

```
Referring to the following document, answer "{response[1]}" in 5 words or less. ←
←
{response[0]} ←
←
Answer:
```

---

Table 12: Prompt templates used in 3-action chains.

# VIST5: An Adaptive, Retrieval-Augmented Language Model for Visualization-oriented Dialog

Henrik Voigt<sup>1</sup>, Nuno Carvalhais<sup>2</sup>, Monique Meuschke<sup>3</sup>,

Markus Reichstein<sup>2</sup>, Sina Zarriß<sup>4</sup>, Kai Lawonn<sup>1</sup>

<sup>1</sup>University of Jena <sup>2</sup>MPI Biogeochemistry <sup>3</sup>University of Magdeburg <sup>4</sup>Bielefeld University

<sup>1</sup>first.last@uni-jena.de, <sup>2</sup>first.last.bgc-jena.mpg.de

<sup>3</sup>last@isg.cs.uni-magdeburg.de, <sup>4</sup>first.last@uni-bielefeld.de

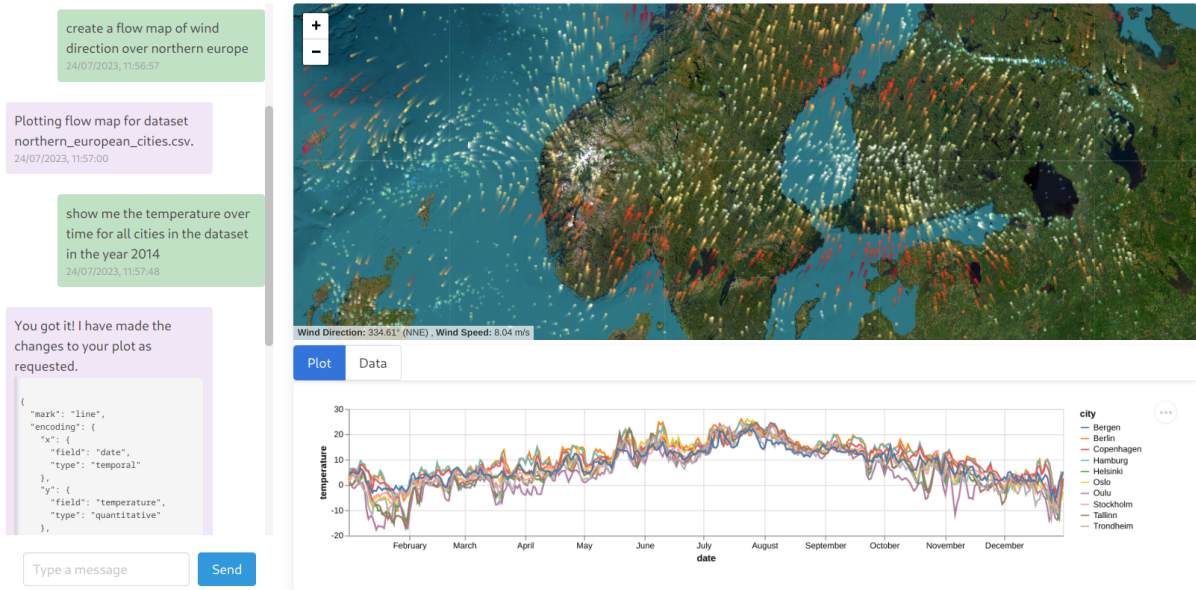


Figure 1: VIST5 makes it easy for researchers and professionals to explore their data using natural language. Users articulate their visualization preferences in a **chat window**, displayed in the left column. The panel lists the responses of the dialog agent, containing both text and custom Vega-Lite visualization code. The right column contains two visualization tools that can be controlled from the chat. At the top, a **geographical map** displays geo-related plots, such as flow visualizations of wind directions. Below is a display area for **Vega-Lite** visualizations that are generated based on user queries to the dataset.

## Abstract

The advent of large language models has brought about new ways of interacting with data intuitively via natural language. In recent years, a variety of visualization systems have explored the use of natural language to create and modify visualizations through visualization-oriented dialog. However, the majority of these systems rely on tailored dialog agents to analyze domain-specific data and operate domain-specific visualization tools and libraries. This is a major challenge when trying to transfer functionalities between dialog interfaces of different visualization applications. To address this issue, we propose VIST5, a visualization-oriented dialog system that focuses on easy adaptability to an application domain as well as easy transferability of language-controllable visualization library functions be-

tween applications. Its architecture is based on a retrieval-augmented T5 language model that leverages few-shot learning capabilities to enable a rapid adaptation of the system.

## 1 Introduction

The field of visualization has witnessed a surge of interest in integrating dialogue interfaces into visualization applications, leading to the development of various visualization-oriented natural language interfaces (V-NLI) (Narechania et al., 2020; Luo et al., 2021b; Liu et al., 2021; Kim et al., 2021). The goal of these systems is to generate visualizations from natural language queries and modify them accordingly in interaction with the user. However, visualization applications exist in various domain contexts, which require specific vocabulary to be parsed and mapped to custom functionalities.

For example, a visualization application that helps researchers analyze climate data will handle different user intent and different, domain-specific visualization libraries than an application in a medical context (Srinivasan et al., 2021; Gao et al., 2015). Certain types of visualization techniques, such as *bar charts*, *line charts*, or *scatter plots*, are very general, so they can be used in almost any domain. Others, such as *flow maps* for the visualization of wind vectors, are not and their access via the natural language interface must be integrated with great effort. Transferring a set of solutions, such as successfully mapping user queries to visualization library functions, from one V-NLI to another without writing new code is still a challenging task. It would be ideal if created functionality could be transferred between V-NLI applications by simply showing the system how to use a particular library with a few examples.

In this paper, we introduce VIST5, a V-NLI that helps users perform text-related visualization tasks while being adaptive to the visualization libraries of the application domain. The system implements a retrieval-augmented language model trained on a mixture of visualization-specific text generation tasks and a large collection of general text-to-text translation tasks. Its retrieval augmentation allows modular extension with domain-specific user commands and portability of functionality between applications. Moreover, the language model meets the requirements of small model size, fast trainability, and fast inference on commodity hardware. We illustrate the adaptation to the specifics of a domain using the example of **climate data exploration**.

Our contributions can be summarized as follows:

- **Efficient Multi-Task Architecture.** Introduction of an efficient and generic multi-task architecture for text-related visualization tasks.
- **Retrieval-Augmented Dialogue System.** The presentation of a dialog system that uses an *information retrieval* component to ground the dialog in knowledge retrieval from external resources. This allows a smaller model size while exploiting knowledge from external databases.
- **Modular Extensibility via Few-Shot Paradigm.** Leveraging the *few-shot* capabilities of the language model to enable modular extensibility and portability of user intents

between applications, as well as integration of new custom intents in minutes.

For a demo video of the VIST5 system please visit <https://youtu.be/bsgaV7hj1Gs>.

## 2 Related Work

Natural language interfaces for data visualization have recently emerged as a powerful combination of visualization and NLP techniques. In their comprehensive survey, Shen et al. (2021) provide an overview of how natural language interaction can be integrated into the visualization pipeline of Card (1999). Voigt et al. (2021, 2022) elaborate on the different visualization tasks that can be facilitated by natural language interactions. The resulting **V-NLI pipeline** is shown in Figure 2. The following is a sequential listing of the steps in the V-NLI pipeline paired with recent work in each step.

**Query Interpretation.** Interpreting the query is about identifying the subset of the *data* the user wants to see and the *actions* the user wants to perform on the data. Setlur et al. (2016) introduced Eviza, which leverages a probabilistic grammar defining a rule-based interaction schema on how to react to specific types of queries. Flowsense (Yu and Silva, 2019), another rule-based semantic parsing approach, matches special utterances and maps them to visualizations in a data flow architecture. Other works focus on resolving linguistic ambiguity and vagueness in expressions using sentiment analysis and word co-occurrence (Hearst et al., 2019; Setlur et al., 2019). Recent systems have introduced neural sequence-to-sequence approaches that translate queries directly into visualizations (Luo et al., 2021b). Maddigan and Susnjak (2023) have conducted an investigation on diverse prompt designs for ChatGPT (Ouyang et al., 2022), OpenAI Codex (Chen et al., 2021), and GPT-3 (Brown et al., 2020), demonstrating the remarkable capability of these LLMs in producing high-fidelity visualizations from natural language input. Our work takes a different approach, considering that training and inferring such large models can be expensive and hardware-intensive, making them unsuitable for computationally constrained use cases. Instead, we concentrate on open access, extensibility, and modularity, offering an alternative perspective.

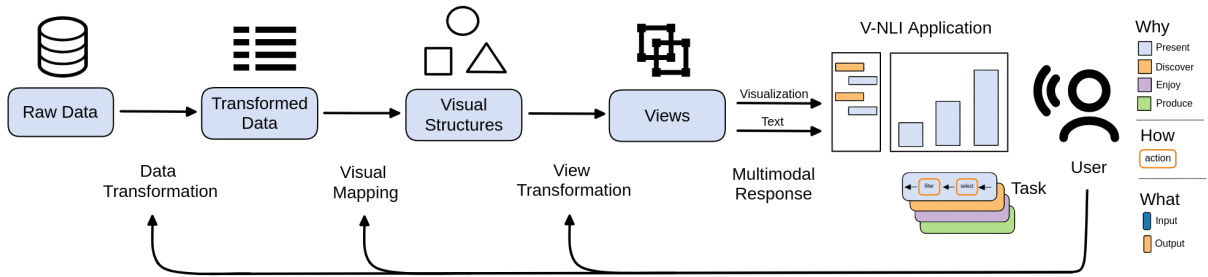


Figure 2: **V-NLI Pipeline.** Given a *user query*, the data is first *transformed*, then *mapped* to visual structures, and then *displayed* in a view. The user, on the other hand, uses the interface by accessing different stages of the pipeline via language to solve a visualization task *action by action*.

**Data Transformation.** Transforming the data according to the action specified by the user is the next step in the V-NLI pipeline (e.g. by *aggregation, filtering, binning, or grouping*). A set of approaches identifies transformation functions from visualization libraries through phrase matching (Gao et al., 2015; Hoque et al., 2017; Sun et al., 2010; Srinivasan and Stasko, 2017; Dhamdhare et al., 2017), others make use of a common data interface such as SQL (Zhong et al., 2017; Wang et al., 2019; Scholak et al., 2021; Xie et al., 2022; Qi et al., 2022).

**Visual Mapping.** In V-NLI systems, the mapping from data to visual representation is usually seen in one of two flavors: 1) the data transformation (e.g. selection of table, column, conditions) and the generation of the visualization specification (e.g. chart type, color) are integrated, as in *ncnet* (Luo et al., 2021b), or 2) the data transformation and visualization specification are separated, with an appropriate visualization for the resulting data being suggested after the query is executed (Wongsuphasawat et al., 2015, 2016; Zhu et al., 2020; Luo et al., 2018). Quda (Fu et al., 2020) and ADVISor (Liu et al., 2021) use neural intent classification methods that are more flexible for integrating custom visualization library functions, but still have the problem of being difficult to extend and adapt to new user intents without retraining.

**View Transformation.** In current systems, manipulation of visual elements in the view is primarily enabled through other channels of multimodal interaction, such as touch and gesture (Kim et al., 2021; Srinivasan et al., 2020b), as exemplified by InChorus (Srinivasan et al., 2020a). Orko (Srinivasan and Stasko, 2017) combines written or spoken text input with touch gestures to manipulate view properties, as does Valletto (Kassel and Rohs, 2018).

### 3 VIST5 System

The VIST5 system is composed of a language model (Section 3.1), a dialog management component (Section 3.2) that controls the memory and API calls to the various visualization libraries used, and a user interface (Section 3.3). The system architecture and the query execution process are shown in Figure 3. The open-source code of the system is available at <https://github.com/clause-bielefeld/VIST5.git>.

#### 3.1 Language Model

The model architecture closely aligns with T5-base and features 12 encoder and decoder blocks with a token embedding dimension of 768 (Raffel et al., 2020). We employ an input context width of 2048 tokens to match the length of the input prompt. Natural language queries are tokenized using the SentencePiece tokenizer from Kudo and Richardson (2018) based on a 32,000 subword vocabulary. In total, this results in a size of 220 million parameters. The model is quantized and deployed in an ONNX runtime, which leads to a small memory footprint of only 225 MB (ONNX Runtime developers, 2018). We initialize with pre-trained FLAN-T5-base (Chung et al., 2022) model weights, which are obtained from the huggingface model hub (Wolf et al., 2020).

**Datasets.** We fine-tune the language model using the following datasets:

- **nvbench.** nvbench is the largest dataset available for the NL2VIS task (Luo et al., 2021a). In nvbench, text queries are translated into Vega-Lite JSON specifications. The dataset contains a large number of 25,750 examples from 750 data tables in 105 domains.

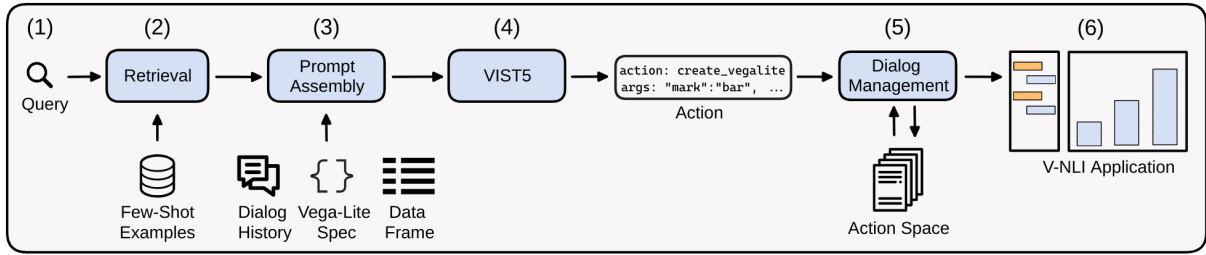


Figure 3: **VIST5 system architecture.** An example query interpretation includes the following steps: 1) The query is tokenized and embedded into a neural embedding vector. 2) The retrieval component returns examples relevant to the query from long-term memory. 3) If similar examples are found, they are included in the prompt along with the *visualization state*, *table state* and *dialog history*. 4) The prompt is fed into the model, which predicts an *action* and *arguments* for that action. 5) The action is validated by the dialog management component and then executed. 6) The output of the action is passed on to the frontend, where it leads to an update of the visualization.

- **Nlv2.** The natural instructions dataset is used for **few-shot instruction** fine-tuning (Wang et al., 2022). The model is trained in such a way that it first sees three similar input/output examples in the prompt before generating a response to the current query. This training objective was explicitly chosen to train the T5 model on cases where **few-shot examples** are available in addition to an input. The goal is to train it to derive a solution (e.g., how to call a particular function) based on given examples and then apply it to the input.
- **Domain-Specific Dialogs.** The VIST5 system is equipped with an online *annotation tool* to capture domain-specific utterances and commands during runtime. We employed it to collect 300 dialog turns from researchers exploring the system. This very small dataset contains contextual queries from the domain of climate science. It is used as a showcase to demonstrate how the annotation tool can be used to adapt the model to a specific domain.

From the above datasets, we use nvbench and the domain-specific dialogs in their entirety. From Nlv2, we take a random sample of 50k. We then use an NVIDIA A6000 GPU to fine-tune the language model for four hours (one epoch).

### 3.2 Dialog Management

To manage the dialog, we use two additional components. The first is the agent’s short-term memory, which stores the *status of the visualization* and the currently selected *data table* as well as the most recent *dialog history*. The second is a long-term memory, which is a vector database of domain-specific *few-shot examples*.

#### 3.2.1 Short Term Memory

The *visualization state* in our application consists of the composition of the currently displayed Vega-Lite chart. This is a JSON object that contains all the properties of the visualization such as *mark* and *channel encodings* as well as *data transformations* like *filters* or *aggregations*. The Vega-Lite JSON object is flattened and converted to a normalized JSON string (Wes McKinney, 2010). The *table state* consists of a Pandas dataframe (pandas development team, 2020), which is serialized as the header, followed by the first three rows. The *dialog history* is stored as a sequence of query/response pairs.

#### 3.2.2 Long Term Memory

The main task of the long-term memory is to adapt the application to the context of use, e.g., domain-specific utterances, libraries, and functions that are used during the analysis of **climate data**. This is realized by storing a list of application-specific few-shot examples. A few-shot example is an input-output pair that contains an example user input and the desired *action*, as well as the *arguments* that the model should use to execute that action. An example to call a function of a domain specific library looks like this: INPUT: show me a heat map of temperature, OUTPUT: action: create\_heat\_map; args: "column": "temperature". During runtime, a Sentence-Transformer (Reimers and Gurevych, 2019) is used to encode the input query into a neural embedding vector. Then, the cosine similarities between the encoded query vector and all stored encoded few-shot example vectors are computed. All examples that exceed a similarity threshold  $\alpha$  are kept. We set  $\alpha$  to a similarity value of 0.8. This ensures that

only very relevant examples are returned. Of the retrieved examples, the top 3 are then passed into the prompt. If no example exceeds the threshold, no example is returned and the model must respond to the input without further assistance based on the knowledge contained in its weights.

### 3.2.3 API Orchestration

To manage the different visualization libraries used, all functionalities (= function names and their arguments in JSON format) are listed in an *action space*. The interpretation of a request from perception to final response is as follows: Upon receiving a user request, relevant examples are first retrieved from long-term memory. The prompt is then assembled from these (potentially) retrieved *few-shot examples*, the current *visualization state* as a Vega-Lite JSON string, the *table state*, and the *user input* (see Appendix A for details). Based on this prompt, the model generates an *action* and the corresponding *arguments*. After generation, the control loop checks to see if the generated action exists in the action space, and if it does, the function is called and executed with the specified arguments. The output of this function is then sent to the frontend, where it causes a change in the targeted visualization display.

## 3.3 User Interface

The user interface is built in HTML, CSS, and JavaScript (see Figure 1). The backend, which serves the website and hosts the language model for inference, is based on fastAPI (tiangolo, 2023). **Visualization Display.** The visualization area consists of a geographic map onto which the climate data is projected. To create the map the visualization library leaflet (leaflet, 2023) is used. Below the map, a display for Vega-Lite visualizations (Satyanarayan et al., 2018) is provided. The visualization is dynamically updated with new visualization specifications generated by the language model based on user requests.

**Chat Window.** On the left side, there is a chat window that contains the dialog history of the conversation. It allows the user to submit requests to the system and view the exact system responses including the generated Vega-Lite specs.

**Online Annotation Tool.** After receiving a response, the user can interactively edit the created Vega-Lite specification if desired. If a customized Vega-Lite specification is to be used as a training example in the future, it can also be immediately

submitted back to the system in this manner.

**Data Display.** The Vega-Lite display can be switched to a data display. It shows an overview of the selected data set with the column headers of the data frame, their data types, and the first 1k rows of the data set.

## 4 Features

The focus of the system is to provide visualizations in response to user queries to help users solve application-specific visualization tasks as defined by Brehmer and Munzner (2013). In the VIST5 system, this involves three main tasks: 1) translating a natural language query into a visualization specification, 2) engaging in a domain-specific analytical conversation by exchanging contextual queries to gain insight into the data, and 3) customizing a visualization specification to meet user needs. To measure the response quality of the system in these tasks, we conducted a user study with 24 participants. It revealed that the system provided high-quality responses to diverse visualization requests, and that the vast majority of few-shot requests were also successful. Of particular note is that the users felt really engaged with the system, as evidenced by the high average number of user turns per dialog of 11.6. A detailed description of the study can be found in Appendix B.

### 4.1 Natural Language Query to Visualization

The Natural Language Query to Visualization (NL2VIS) task is the most prominent task supported by the system (Luo et al., 2021a). Given a query, the system responds with a Vega-Lite specification that it believes is the best one to help users answer their question. To demonstrate, consider the query: "Show me Seattle's temperature in 2018 as a line chart". The query is entered into the dialog interface and sent to the backend. Since the model was trained on this task, there are no few-shot examples stored in long-term memory for it. As a result, no examples are added to the prompt. The prompt is then fed to the model. The model recognizes the NL2VIS request and generates a `create_vegalite` action with the appropriate arguments `"mark": "line", "encoding_x_field": "date", "encoding_x_type": "temporal", ...`. The generated specification is then converted from a normalized JSON string back to a JSON object, passed to the front end, and displayed to the user.

## 4.2 Analytical Conversations

Analytic conversations, consisting of a back-and-forth of contextual queries and responses, are critical because, in data exploration, no one knows where insights will be found until they see the data. Often, interest in certain aspects of the data is highly situational, leading to contextual queries. For example, a user might first query the temperature in Seattle, as in the previous example. After viewing the output, the user is interested in comparing this temperature curve to the city of New York, which is on the other side of the continent. In this context, given the initial visualization, the user might simply ask, "Okay, now add the temperature in New York to the plot. This request implies to the model that 1) the user wants to keep the temperature in Seattle in the plot, 2) the user wants to add the temperature in New York to the plot, 3) the year of focus is 2018, and 4) it might be better to color the curves for the two cities differently, otherwise it will be difficult for the user to compare the two. Extending a language interface from single-turn interactions, such as NL2VIS queries, to contextual queries greatly increases its flexibility, since practical use is always contextual.

**Visualization Customization.** Since the Vega-Lite specifications are available to the model in the prompt, users can also customize data-only visualizations by adding titles, labels, changing colors, or swapping axes on the fly. After completing their exploration, users may want to share a plot with their colleagues to discuss an interesting trend in the temperature curves for New York and Seattle that they observed during the exploration. To accomplish this, a user could give the instruction: "Add a title to the chart that reads Seattle vs. New York Temperature 2018". The model will update the plot, and once received, the user can share the visualization with a colleague.

**Domain-Specific Visualizations.** The analysis of climate data depends heavily on the interpretation of the measurements in the context of the geographical location of a weather station. Only when the characteristics of the environment in terms of altitude, vegetation, and urbanization can be considered together with the data, reliable conclusions can be drawn. To this end, we integrate three geospecific plot types to expand the range of options available to climatologists working with VIST5. For example, we enable *marker plots* of weather stations on the leaflet map, giving the user an

overview of where weather stations are located. A second function is the generation of *heat maps*, which can be specified by naming the column in the dataset from which a heat map is to be generated. An example would be "Show me a heat map of precipitation". This is an instruction that the model has never seen during training, but it can be solved by seeing a few examples. The third geospatial map we have integrated following this paradigm is *flow maps* to visualize wind directions. **Custom Functionalities.** Custom functionalities are functions that are provided by the application but usually have to be integrated into the language interface by hand, otherwise, they are inaccessible without training data. Using the few-shot paradigm, we integrate a function to *export* plots and share them with colleagues. Furthermore, it is possible to change the *map type* between satellite/dark/street/hybrid, depending on the interest of the exploration scenario. Finally, it is also possible to ask the model to *update* the weather dataset with fresh data points from the Open Meteo Weather API ([open meteo, 2023](#)). When exploring climate data on maps, it is particularly helpful to use large screens. A drawback for the language interface, in this case, is that typing-based chat is very impractical, as it is annoying to switch back and forth between the keyboard and the screen. We, therefore, decided to include a number of voice locomotion interactions in the form of few-shot examples. We use a text-to-speech service based on the VOSK library ([Shmyrev and other contributors, 2022](#)). Interactions include *zoom in/out*, *move left/right/up/down*, and *navigating* to a specific location by naming it as in "Navigate to the city of London, please.". The map adjusts seamlessly and exploration can continue hands-free.

## 5 Conclusion

In this work, we have proposed VIST5, a system that demonstrates the adaptation of a V-NLI to an application domain using online annotation and few-shot learning techniques. The system performs a retrieval-augmented dialog by using the external knowledge contained in few-shot examples to generate responses to user input. This makes it fast, modular, and easily adaptable to a user-defined domain. Unlike large language models, VIST5 focuses on small model size, fast trainability, and fast inference on commodity hardware to meet the needs of applications with privacy concerns or lim-

ited computational resources. We hope that the system will inspire the community to further improve the architecture and create more applications and datasets for visualization-oriented dialogue to promote the combination of NLP and visualization techniques.

## Limitations

Compared to very large models such as GPT-4, PaLM2, or ChatGPT, VIST5’s capabilities are limited to a much smaller set of tasks. The model is not a general dialog agent like, e.g., ChatGPT and only works on tasks for which it has been trained, or if it is provided with sufficient few-shot examples by the retrieval mechanism. We see this limitation as a clear trade-off that the application developer has to make between the size of the model that can be used in their application and the model properties that are needed for the current application.

A second limitation we see is the collision of similar few-shot examples when the number of tasks to be integrated via the few-shot paradigm becomes very large. This can lead to the retrieval mechanism not always returning the optimal examples and thus providing the model with incorrect starting points that reduce the response quality. A possible compromise here could be to fine-tune the sentence transformer model on the large set of few-shot examples to ensure that the optimal examples are always retrieved.

A third limitation we see is the limitation of the model to generate complete visualization specifications only from the Vega-Lite visualization library. Adding functionality from other visualization libraries such as D3.js or Observable Plot is possible via the few-shot paradigm, but the longer the visualization specifications to be generated, the more error-prone the few-shot approach becomes for small models such as T5-base (e.g., large Vega-Lite specifications can contain more than a hundred properties). We see three approaches as promising directions for the future: 1) visualization specifications for general plots, e.g. bar charts, are specified in a library-independent way and can then be parsed from the general specification into the respective library, 2) methods for integrating code documentation of specific libraries into the prompt and making it usable so that even small language models can benefit from it need to be explored, 3) for large plot specifications of specific visualization libraries, training data needs to be generated either

by humans or (depending on quality requirements) by larger models, e.g. GPT-4.

## Ethics Statement

The nvbench and Niv2 datasets, as well as the T5 and FLAN-T5 models, are available for research and non-commercial use. We explicitly state that the intended use of our model is to assist researchers and domain experts in their data exploration procedures by allowing them to easily generate visualizations from natural language descriptions. The reliability of the generated visualizations and their one-to-one correspondence with the underlying data set must always be verified by the user of the VIST5 system. The language model generates visualizations based on the input query and the information contained in the prompt, within its capabilities. During generation, misinterpretations or misapplied data transformations may occur, leading to incorrect results. Therefore, we encourage users not to take the results generated by the model for granted, but to verify the generation process by always double-checking the specifications provided in the chat window for the generated visualizations and making sure that they make sense in the current context given the query and dataset at hand.

## Acknowledgments

This work was supported by the Carl Zeiss Foundation in the context of the "A Virtual Workshop for Digitization in the Sciences" and "Interactive Inference" projects.

## References

- Robert Amar, James Eagan, and John Stasko. 2005. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 111–117. IEEE.
- Matthew Brehmer and Tamara Munzner. 2013. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*, 19(12):2376–2385.
- 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.



- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz—a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.
- Mackinlay Card. 1999. *Readings in information visualization: using vision to think*. Morgan Kaufmann.
- 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*.
- 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*.
- Kedar Dhamdhere, Kevin S McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. Analyza: Exploring data with conversation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 493–504.
- Siwei Fu, Kai Xiong, Xiaodong Ge, Siliang Tang, Wei Chen, and Yingcai Wu. 2020. Quda: natural language queries for visual data analytics. *arXiv preprint arXiv:2005.03257*.
- Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th annual acm symposium on user interface software & technology*, pages 489–500.
- Marti Hearst, Melanie Tory, and Vidya Setlur. 2019. Toward interface defaults for vague modifiers in natural language interfaces for visual analysis. In *2019 IEEE Visualization Conference (VIS)*, pages 21–25. IEEE.
- Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics*, 24(1):309–318.
- Jan-Frederik Kassel and Michael Rohs. 2018. Valletto: A multimodal interface for ubiquitous visual analytics. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6.
- Young-Ho Kim, Bongshin Lee, Arjun Srinivasan, and Eun Kyoung Choe. 2021. Data@ hand: Fostering visual exploration of personal data on smartphones leveraging speech and touch interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–17.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- leaflet. 2023. [Leaflet/leaflet](#).
- Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE.
- Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. Deepeye: Towards automatic data visualization. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 101–112. IEEE.
- Yuyu Luo, Jiawei Tang, and Guoliang Li. 2021a. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*.
- Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2021b. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226.
- Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. *arXiv preprint arXiv:2302.02094*.
- Arpit Narechania, Arjun Srinivasan, and John Stasko. 2020. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.
- ONNX Runtime developers. 2018. [ONNX Runtime](#). <https://onnxruntime.ai>.
- open meteo. 2023. [open-meteo/open-meteo](#).
- 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*.
- The pandas development team. 2020. [pandas-dev/pandas: Pandas](#).
- Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.
- 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.

- Nils Reimers and Iryna Gurevych. 2019. [Sentence-bert: Sentence embeddings using siamese bert-networks](#). *CoRR*, abs/1908.10084.
- Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2018. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23:341–350.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*.
- Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. 2016. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology*, pages 365–377.
- Vidya Setlur, Melanie Tory, and Alex Djalali. 2019. Inferring underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 40–51.
- Leixian Shen, Enya Shen, Yuyu Luo, Xiacong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2021. Towards natural language interfaces for data visualization: A survey. *arXiv preprint arXiv:2109.03506*.
- Nickolay V. Shmyrev and other contributors. 2022. Vosk Speech Recognition Toolkit: Offline speech recognition API for Android, iOS, Raspberry Pi and servers with Python, Java, C and Node. <https://github.com/alphacep/vosk-api>.
- Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M Drucker, and Ken Hinckley. 2020a. Inchorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13.
- Arjun Srinivasan, Bongshin Lee, and John Stasko. 2020b. Interweaving multimodal interaction with flexible unit visualizations for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 27(8):3519–3533.
- Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M Drucker, and John Stasko. 2021. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.
- Arjun Srinivasan and John Stasko. 2017. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE transactions on visualization and computer graphics*, 24(1):511–521.
- Yiwen Sun, Jason Leigh, Andrew Johnson, and Sangyoon Lee. 2010. Articulate: A semi-automated model for translating natural language queries into meaningful visualizations. In *Smart Graphics: 10th International Symposium on Smart Graphics, Banff, Canada, June 24-26, 2010 Proceedings 10*, pages 184–195. Springer.
- tiangolo. 2023. [tiangolo/fastapi](#).
- Henrik Voigt, Özge Alaçam, Monique Meuschke, Kai Lawonn, and Sina Zarrieß. 2022. The why and the how: A survey on natural language interaction in visualization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 348–374.
- Henrik Voigt, Monique Meuschke, Kai Lawonn, and Sina Zarrieß. 2021. Challenges in designing natural language interfaces for complex visual models. In *Proceedings of the First Workshop on Bridging Human-Computer Interaction and Natural Language Processing*, pages 66–73.
- Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoor-molabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: generalization via declarative instructions on 1600+ tasks. In *EMNLP*.
- Wes McKinney. 2010. [Data Structures for Statistical Computing in Python](#). In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.
- 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.
- Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658.
- Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–6.
- Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang,

et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Bowen Yu and Cláudio T Silva. 2019. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics*, 26(1):1–11.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Sujia Zhu, Guodao Sun, Qi Jiang, Meng Zha, and Ronghua Liang. 2020. A survey on automatic infographics and visualization recommendations. *Vis. Informatics*, 4:24–40.

## A Prompt Design

The prompt is assembled as a sequence of the *visualization state* and the *table state*. Below that we put the *dialog history*, followed by a new line signaling the new input *query*. After the input query, relevant examples from long-term memory are displayed. A visual summary of the prompt design can be seen in Figure 4.

```
Visualization State:
"mark": "bar", "encoding_x_field": "city", "encoding_x_type": "nominal",
"encoding_y_aggregate": "mean", "encoding_y_field": "temperature",
"encoding_y_type": "quantitative"

Table State:
table_name : northern_european_cities
col : date (object) | temperature (float64) | radiation (float64) ...
row_0 : 2003-02-16 | -3.5 | 4.34 ...

=====

Conversation History:
INPUT: Hello
OUTPUT: action: text_response; args: "text": "Hi, how can I help?"
INPUT: show me a bar chart of the mean temperature per city
OUTPUT: action: create_vegalite; args: "mark": "bar", "encoding_x_field": "city", ...

NEW INPUT: change the map type to hybrid please

Here are some examples:
INPUT: change the leaflet map type to street
OUTPUT: action: change_map; args: "type": "street"
#
INPUT: change map type to satellite
OUTPUT: action: change_map; args: "type": "satellite"
#
INPUT: turn the leaflet map theme to dark
OUTPUT: action: change_map; args: "type": "dark"

OUTPUT: action: change_map; args: "type": "hybrid"
```

Figure 4: **Example prompt of the VIST5 language model.** *Blue*: The visualization state contains the stringified Vega-Lite specification. *Black*: The table state contains a stringified version of the column header and the first three rows of the Pandas data frame of the currently used dataset. *Green*: The conversation history contains up to eight previous turns in the dialog. *Red*: The new input field contains the current user query. *Purple*: The examples section contains up to three possible retrieved few-shot examples from long-term memory. *Orange*: The word OUTPUT is the last word entered into the model, signaling the start of the generation process. The subsequent action and arguments are possible outputs to be generated by the model given the preceding prompt.

## B Evaluation

We evaluated the system by conducting an active user study engaging 24 users with the VIST5 dialog assistant. The user study was conducted with people of academic background (58.3% male, 37.5% female, 4.2% prefer not to say). 8.4% of the participants are in NLP, 54.2% are in Visualization, 20.8% are in climate science, and 16.6% are people from other fields subsumed under 'Others'. 62.5% of the participants were between the ages of 20 and 30, 29.2% were between 30 and 40, and 8.3% were between 40 and 50. 29.2% had less than three years of experience in their domain, 37.5% between three and five years, and 33.3% more than five years.

### B.1 Method

The main goal of our study was to find out:

1. The quality of the answers given by the system with respect to the different types of queries in the **NL2VIS task**.
2. The system's response quality on **few-shot tasks**.

We put participants into a task-oriented dialog situation. Users were given the option to choose from a set of *seven* different climate data sets. To generate goals for users to achieve with the system, we generate visualization tasks from the pool of common low-level visualization tasks specified by Amar et al. (2005): *characterize distribution, compute derived value, correlate, determine range, filter, find extremum, find anomalies, cluster, retrieve value, sort*. Every user is randomly assigned **two** of those tasks. A low-level visualization task is presented to the user as a general instruction, e.g., to filter the dataset according to a certain condition. The user must then try to solve the task by interacting with the chatbot. Further, every participant was assigned **one** few-shot task from the pool of few-shot categories: *custom visualization, custom functionality, locomotion* which each is comprised of several few-shot tasks, but we are mainly interested in the response quality per category. The custom visualizations that can be created are *marker plots, heat maps, flow visualizations*. Custom functions to be invoked include *exporting visualizations, changing map style, and updating the dataset*. Locomotion few shot tasks include *zooming in/out, moving left/right/up/down, and navigating* to a city of choice. To solve a task, a user can ask as many questions as necessary. During the interaction, users are prompted to rate the quality of each response from the chatbot on a Likert scale from 1 (poor) to 5 (very good), i.e. how appropriate the response was given the query. In addition, users are asked to provide textual feedback on what they consider to be particularly good or bad answers. This helps us understand these extreme cases better in hindsight and learn from them. Before the study began, users were shown a video of a short sample conversation (less than 10 turns) between a user and the chatbot, explaining how to rate responses and where to provide feedback.

Once all tasks have been completed, we allow the participants to explore the system freely in an

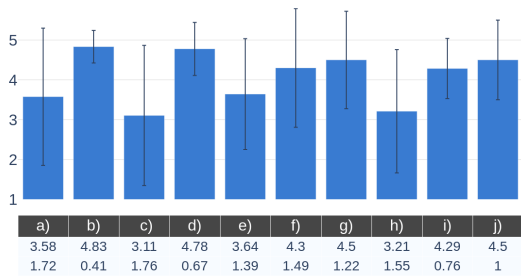


Figure 5: Results of the user evaluation on the ten low-level tasks of Amar et al. (2005): a) *characterize the distribution*, b) *compute a derived value*, c) *correlate*, d) *determine range*, e) *filter*, f) *find extremum*, g) *find anomalies*, h) *cluster*, i) *retrieve value*, j) *sort*. The mean is provided in the first row of the table below, *std* in the second.

unbounded way. The unconstrained interaction helps us get additional feedback for a broader horizon of uses that we may not have thought of before. This feedback is interesting for guiding future work.

## B.2 Results

All in all, we collected a set of 279 dialog turns from the users during the study. The average dialog has a number of 11.6 user turns, which is higher than the average number of user turns in current task-oriented dialog datasets such as MultiWOZ (Budzianowski et al., 2018).

**NL2VIS Tasks.** The results on the low-level visualization tasks are shown in Figure 5. The mean Likert score across all tasks is 3.82. The standard deviation across all tasks is 1.53. The mean for each task is shown in the first row of the table in Figure 5, and the standard deviation is shown in the second row. We can see that the mean score for the tasks *compute derived value*, *determine range*, *find extremum*, *find anomalies*, *retrieve value* and *sort* is very high, with an average value above 4. This tells us that the system provides high-quality responses for these subsets of low-level visualization tasks.

Tasks like *characterize distribution*, *correlate*, *filter* and *cluster* have an average value above 3, but also show a larger standard deviation. This shows that for these tasks the response quality varies more between appropriate and inappropriate responses, but the tendency is towards positive responses. Overall, the system does not perform below average on any of the tasks.

**Few-Shot Tasks.** The results on the few-shot tasks are shown in Figure 6. The average rating over all

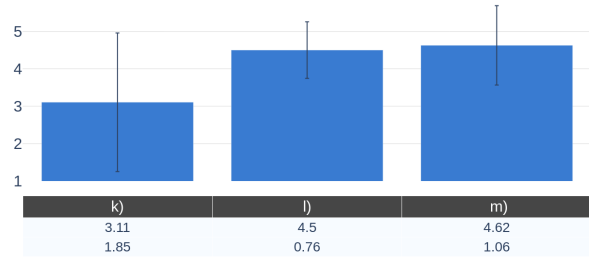


Figure 6: Results of the user evaluation on the three few-shot task categories: k) *custom functionality*, l) *custom visualization*, m) *locomotion*. The mean is provided in the first row of the table below, *std* in the second.

tasks is 3.77. The standard deviation over all tasks is 1.65. The mean for each task is shown in the first row of the table in Figure 6, and the standard deviation is shown in the second row. We can see that the means for the *custom visualization* task and the *locomotion* task are very high with values above 4. This shows that the system had no problems finding out how to create custom visualizations on the leaflet map and navigating it based on a few examples. The mean scores for the *custom functionality* task are above 3 and show higher standard deviations, indicating that the response quality is more variable for this few-shot category. We found a possible explanation for this in the vulnerability of the few-shot paradigm to typos. In particular, typos when changing the map type or selecting column names cause problems because the system usually passes the arguments as they are given in the input to the function, which then leads to errors in execution. The integration of a spell checker or the use of system-initiated check questions in case of uncertainty are possible levers for future improvements in this respect.

Overall, the system always scores above the mean of 3 for all tasks. This shows that, on average, users found the responses to be helpful. However, it also shows that while the system performed well on the majority of responses, it did not perform optimally on all inputs.

Arno Candel, Jon McKinney, Philipp Singer, Pascal Pfeiffer,  
Maximilian Jeblick, Chun Ming Lee, Marcos V. Conde

H2O.ai, Inc.

Mountain View, CA

{firstname.surname}@h2o.ai

<https://gpt.h2o.ai/>

## Abstract

Large Language Models (LLMs) represent a revolution in AI. However, they also pose many significant risks, such as the presence of biased, private, copyrighted or harmful text. For this reason we need open, transparent and safe solutions. We introduce a complete open-source ecosystem for developing and testing LLMs. The goal of this project is to boost open alternatives to closed-source approaches. We release h2oGPT, a family of fine-tuned LLMs from 7 to 70 Billion parameters. We also introduce H2O LLM Studio, a framework and no-code GUI designed for efficient fine-tuning, evaluation, and deployment of LLMs using the most recent state-of-the-art techniques. Our code and models are licensed under fully permissive Apache 2.0 licenses. We believe open-source language models help to boost AI development and make it more accessible and trustworthy.

## 1 Introduction

Since the Transformer (Vaswani et al., 2017) was introduced in the Natural Language Processing (NLP) community, the advances in this field have increased exponentially (Wolf et al., 2020).

Starting from popular models such as BERT (Devlin et al., 2018a) or Generative Pre-trained Transformers (GPT) (Radford et al., 2018) -both introduced in 2018-, researchers have been pushing the limits of scaling and learned representations in language models (Liu et al., 2019; Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022).

Recent advances in Large Language Models (LLMs) are all over the news; these models represent a revolution in Artificial Intelligence (AI) due to their real-world applications through natural language processing (NLP), from internet chatbots to virtual assistants and programmers. However, these also pose significant risks and challenges. The most popular models (e.g., chatGPT (OpenAI, 2023)) are proprietary and not truly open-source, either transparent regarding their training data.

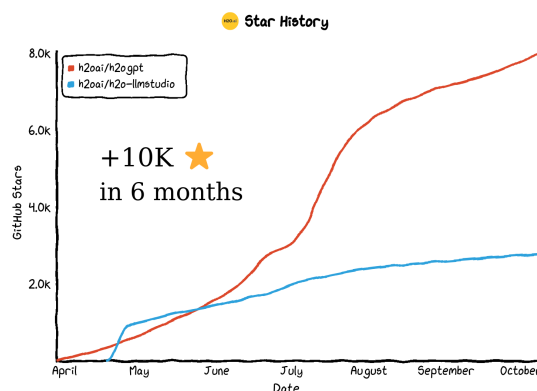


Figure 1: Evolution of our project in GitHub. Our tools have been widely adopted by the NLP community. See <https://github.com/h2oai/h2ogpt>.

This fast advance leads to a wide range of practical challenges that must be addressed in order for these models to be widely utilized and explored. The popularity and demand of LLMs call for systems to train, fine-tune, evaluate, scale, and deploy the models on a variety of platforms. Given the training costs (millions of dollars), practitioners increasingly rely on pre-trained general-purpose LLMs and fine-tune them for specific downstream tasks and datasets. This requires a wide catalogue of open-source pre-trained LLMs, and sophisticated procedures and tools for efficient fine-tuning. Moreover, considering the massive size of these models (usually from 7 to 100 Billion parameters), we also need compression techniques to deploy them successfully on different platforms.

We believe open-source language models help to boost AI development and make it more accessible and trustworthy. They lower entry hurdles, allowing people to tailor these models to their needs. This openness increases innovation, transparency, and fairness. As part of this effort, we **introduce two open-source libraries**: *h2oGPT* and *H2O LLM Studio*, for LLMs development, including Multi LLM deployment and evaluation — widely adopted in the NLP community (see Fig. 1).

**h2oGPT** (<https://github.com/h2oai/h2ogpt>) is a library dedicated to supporting open-source LLMs research, and facilitating their integration while ensuring privacy and transparency. Most integrated models are designed for both research and production. The main use-case of this library is to deploy and test efficiently a wide variety of LLMs on private databases and documents. This tool allows users to compare different models on several tasks and datasets concurrently. An example of this application is <https://gpt.h2o.ai/>.

**H2O LLM Studio** (<https://github.com/h2oai/h2o-llmstudio>) complements the previous library, and allows users to efficiently fine-tune any LLM using the most recent *state-of-the-art* techniques such as LoRA adapters (Hu et al., 2021), reinforcement learning (RLHF), and 4-bit training. After fine-tuning (or training), the models can be easily exported and deployed at the Hugging Face Hub <sup>1</sup>. Moreover, the library includes a graphic user interface (GUI) specially designed for large language models.

*h2oGPT* and *H2O LLM Studio* are an ongoing effort maintained frequently by the team of engineers and researchers at H2O.ai with exciting support from the open-source NLP community and external contributors. Both are released under the Apache 2.0 license <sup>2</sup>. Tutorials and detailed documentation are available at the corresponding websites and the technical report (Candel et al., 2023).

## 2 Related Work

Large language models (LLMs) are designed to process and understand vast amounts of natural language data *e.g.*, internet questions, text in documents, financial data, textbook material, etc. As foundation models (Bommasani et al., 2021), these are trained from broad data at scale (Howard and Ruder, 2018), and can be adapted (*ie.* fine-tuned) to a wide range of down-stream tasks (Wang et al., 2018; Lewis et al., 2019).

They are built on the *Transformer* neural network architecture (Vaswani et al., 2017), which allows them to capture complex language patterns and relationships. Derived from the *Transformer*, we find BERT-like models (Devlin et al., 2018b; Le et al., 2020; Liu et al., 2019) focused on pre-training with bidirectional encoders. We also find

the popular Generative Pre-trained Transformers (GPTs) (Radford et al., 2018, 2019; Brown et al., 2020; OpenAI, 2023), focused on generative pre-training. These serve as the engine of chatGPT.

Since 2022, we experience a new revolution in NLP with the rise of LLMs (over billion parameters models). These models usually follow a multi-stage training strategy, starting with a task-agnostic pre-training on large and diverse datasets. Some related LLMs are LLaMA (Touvron et al., 2023a), GPT-NeoX (Black et al., 2022), BLOOM (Scao et al., 2022), Palm (Chowdhery et al., 2022), OPT (Zhang et al., 2022), and GPT-4 (OpenAI, 2023). We also explore community models such as Falcon (Penedo et al.), Alpaca (Taori et al., 2023), and OpenAssistant (Köpf et al., 2023).

### 2.1 Why Open-Source LLMs?

While commercially hosted and centralized LLMs like ChatGPT -based on GPT-4 (OpenAI, 2023)-, Microsoft's Bing AI Chat, and Google's Bard are powerful and effective, they have certain risks and limitations compared to open-source LLMs:

- **Data Privacy and Security:** Many require sending data to external servers. This can raise concerns about data privacy, security, and compliance, especially for sensitive information or industries with strict regulations.
- **Dependency and Customization:** We want to allow users to train LLMs on private data safely, and customize the models to their specific needs and applications. Moreover the users can deploy them on their own infrastructure, and even modify the underlying code.
- **Traceability and Transparency:** To understand the risky behaviours of LLMs (*e.g.*, hallucinations, biases, private information etc.), and ensure their safe and trustworthy use, it is fundamental to analyze the dataset and training strategies used to produce such model.
- **Carbon footprint:** Users tend to adopt our open *state-of-the-art* models, instead of running expensive and complicated experiments (in most cases to replicate results). Therefore, we aim to reduce the overall carbon footprint (*ie.* GPU hours consumption) by providing high-quality models and tools.

Overall, open-source LLMs offer greater flexibility, control, and cost-effectiveness, while addressing data privacy and security concerns.

<sup>1</sup><https://huggingface.co/models>

<sup>2</sup><https://www.apache.org/licenses/LICENSE-2.0>

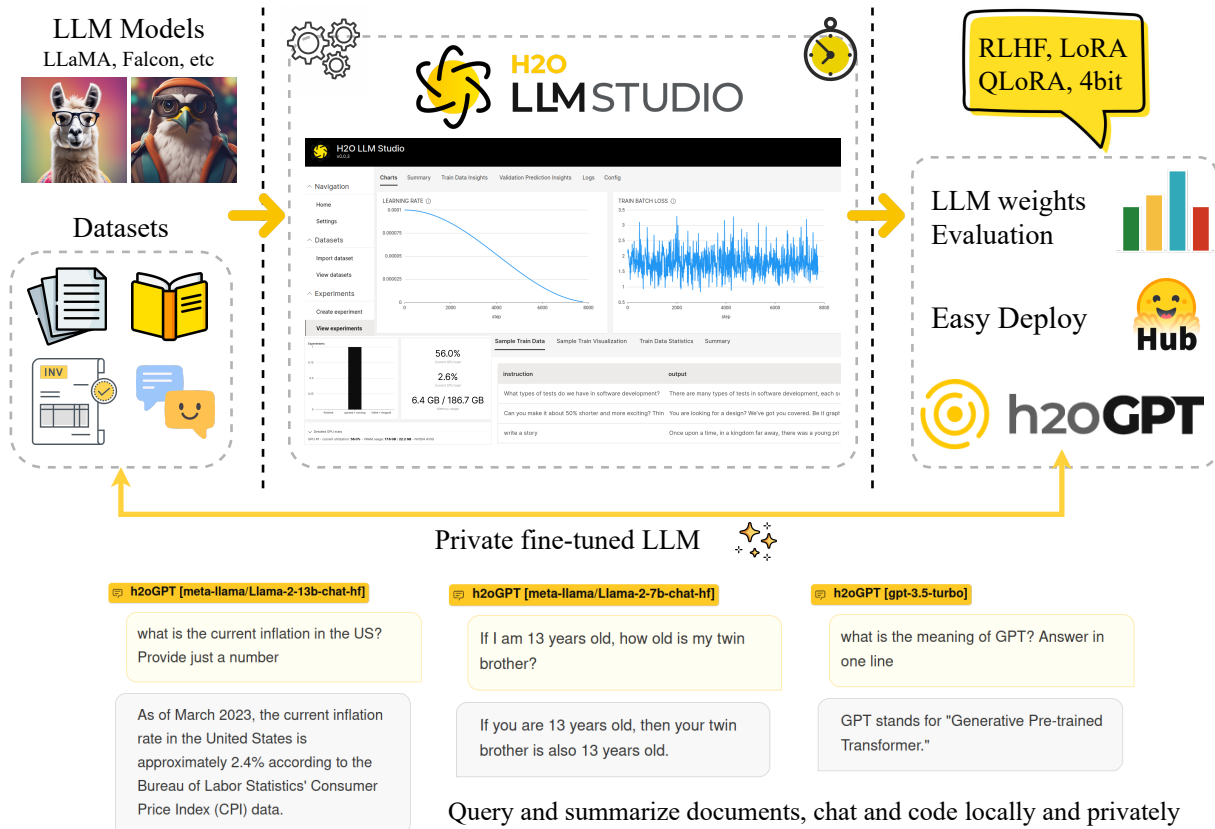


Figure 2: **Open LLM Ecosystem.** (left) The user does not need to transfer private data to 3rd parties, and can select any popular LLM *e.g.*, LLaMA, Falcon. (mid) H2O LLM Studio allows to train and fine-tune any language model using *state-of-the-art* techniques and a GUI without coding. (right) The models can be easily evaluated, exported and deployed. More information at <https://github.com/h2oai/h2o-llmstudio>. Apache 2 License.

### 3 H2O LLM Studio

An open-source framework for efficient fine-tuning LLMs without coding, using a graphic user interface (GUI) specially designed for large language models<sup>3</sup>. This is illustrated in Figures 2 and 4.

We use the most popular **adapters** for fast fine-tuning such as Low-Rank Adaptation (LoRA) (Hu et al., 2021) and QLoRA (Dettmers et al., 2023), as well as 8-bit (up to 4-bit) model training with a low memory footprint, and the corresponding **quantization**. This allows to fine-tune small LLMs in regular GPUs, even using Google Colab or Kaggle. For example < 10B models (*e.g.*, LLaMa-2 7B) can be fine-tuned in a single NVIDIA-T4 (16GB).

We also integrate *Reinforcement Learning from Human Feedback (RLHF)* (Ouyang et al., 2022; Stiennon et al., 2020). This feature is inspired in TRL<sup>4</sup> (von Werra et al., 2020), with the Proximal Policy Optimisation (PPO) by (Ziegler et al., 2019).

<sup>3</sup><https://github.com/h2oai/h2o-llmstudio>

<sup>4</sup><https://github.com/lwerra/trl>

LLM Studio allows complete **customization** of the experimental setup: dataset, *state-of-the-art* model selection, optimizer, learning rate schedule, tokenizer, sequence length (number of tokens), low-rank adapter, validation set and metrics, etc.

The users can **track** several simultaneous experiments, and easily **export** the logs and results. Moreover, the models can be easily exported to the Hugging Face Hub, to be shared with the community or deploy locally and privately.

The framework supports **any open-source language model**, we here highlight the most popular *state-of-the-art* large models: GPT-NeoX (Black et al., 2022), Falcon (Penedo et al.), LLaMa and Llama 2 (Touvron et al., 2023b), Vicuna (Chiang et al., 2023), WizardLM (Xu et al., 2023; Luo et al., 2023), h2oGPT (Candel et al., 2023), and MPT (MosaicML, 2023). We summarize these models in Table 1. Most models are trained on a large amount of data (over 1T tokens), they can handle extremely long inputs (large context length), and are licensed for commercial use.



| Model                             | Size (B)     |
|-----------------------------------|--------------|
| Llama 2 (Touvron et al., 2023b)   | 7 / 13 / 70  |
| CodeLlama (Touvron et al., 2023b) | 34           |
| Falcon (Penedo et al.)            | 7 / 40 / 180 |
| Mistral AI (Mistral AI, 2023)     | 7            |
| GPT-NeoX (Black et al., 2022)     | 20           |
| WizardLM (Xu et al., 2023)        | 7 / 13 / 70  |
| Vicuna (Chiang et al., 2023)      | 13           |
| MPT (MosaicML, 2023)              | 7 / 30       |
| h2oGPT (Candel et al., 2023)      | 7 to 70      |
| GPT-3.5 (by OpenAI)               | ?            |

Table 1: Most popular pre-trained LLMs for fine-tuning. We report the size in Billions (B) of parameters.

We acknowledge **other existing tools** such as LLMTune (Kuleshov, 2023) and EasyLM (Geng, 2023). However, these do not include as many features as LLM Studio (*e.g.*, GUI, supported models and techniques, etc), their licenses can be less permissive. Our tools are amongst the most adopted LLM-related software in GitHub (considering stars and forks by July 2023) — see Fig. 1.

## 4 Multi LLM Deployment and Evaluation

Any model produced from LLM Studio can be easily integrated into HuggingFace’s space & models. We refer to our own space for more information and access to our models <sup>5</sup>.

In Fig. 3 (top) we show a snapshot of our demo h2oGPT <https://gpt.h2o.ai/>. We deploy multiple *state-of-the-art* LLM models including Falcon (7/40B), Llama 2 (7/13/70B), and GPT-3.5. This allows us to compare different models and setups.

The user’s prompt is evaluated by the different LLMs **concurrently**. We can see the answer generation progress for each model, at the same time. Using this software we can identify clear differences between LLMs easily, for example fast/low inference, hallucinations, common response patterns, bias, memorized data etc. Also, we can analyze the effect of **prompt engineering** on the different models and expose vulnerabilities. The users can deploy the models on a wide variety of inference servers (HF TGI server, vLLM, Gradio, OpenAI), and evaluate performance using reward models.

**Document Analysis** *h2oGPT* also allows to query and summarize documents in many formats (*e.g.*, PDFs, Word, Code, Text, Markdown, etc).

<sup>5</sup><https://huggingface.co/h2oai>

We implement an efficient use of context using instruct-tuned LLMs (no need for LangChain).

Note that this ecosystem can be reproduced locally, to analyze the models in a private and safe manner. We also provide a OpenAI-compliant Python client API for client-server control.

**Guides & Material** We provide a short **Video tutorial (2 mins)**, and a complete **video overview** of the ecosystem (16 min, 340K views) on YouTube.

Also a step-by-step tutorial **Make Your Own GPT With h2oGPT & H2O LLM Studio** (1hr).

We also host all of our models in HF: <https://huggingface.co/h2oai>. We refer the reader to our GitHubs for more demos, and documentation.

## 5 Future Work

Our open-source LLM Ecosystem is in constant development, *h2oGPT* and *LLM Studio* are updated based on the most recent research advances and demands. We plan to integrate new model quantization techniques, distillation and long-context training (context length over 100K tokens).

We also plan to support more multi-lingual models, and multi-modal models.

## 6 Limitations

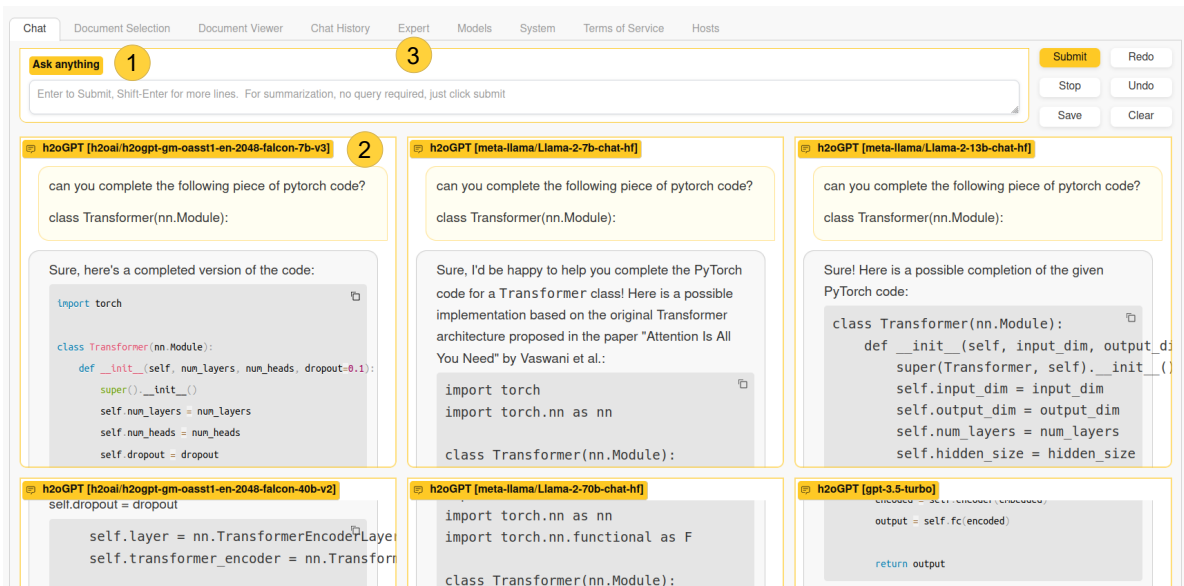
**Datasets** Fine-tuning requires data text pairs of instruction and expected result/answer.

**Biases and Offensiveness** LLMs are trained on a diverse range of unfiltered internet text data, which may contain biased, racist, offensive, or otherwise inappropriate content. Therefore, the generated content by these models may sometimes exhibit biases or produce content that is offensive or inappropriate. We do not endorse, support, or promote any such content or viewpoints.

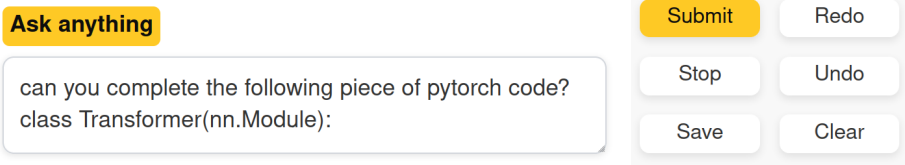
**Usage** The large language model is an AI-based tool and not a human. It may produce incorrect, offensive, nonsensical, or irrelevant responses. It is the user’s responsibility to critically evaluate the generated content and use it at their discretion.

**Carbon footprint** Training LLMs is expensive and their use is associated to tons of CO<sub>2</sub> emissions (Touvron et al., 2023a).

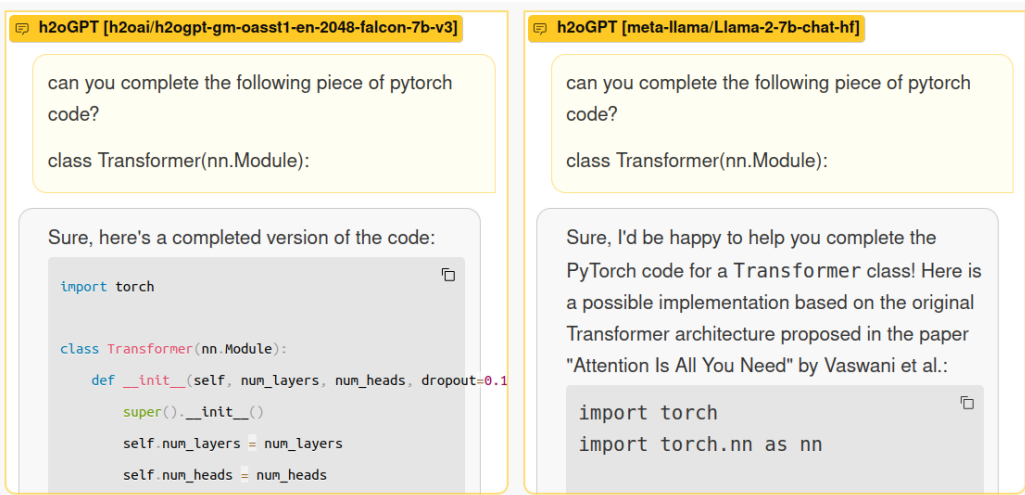
**Hallucinations** LLMs are probabilistic, therefore, certain “random” behaviour is natural and expected, especially on complex prompts (*e.g.*, logical paradoxes, reasoning problems, etc) and “unknown content” not present in the training corpus.



- 1 **Input prompt.** The users clicks on *submit* and the multiple LLMs will start to interact. You can also *save* the prompt, stop execution, etc.



- 2 **Multiple LLM evaluation.** This visualization-evaluation allows the user to detect clear differences between the models for example, inference speed and clear hallucinations.



- 3 **Expert mode.** Users can change the *temperature*, cumulative probabilities (*top p*), context (*top k tokens*), maximum output length, maximum runtime, etc.

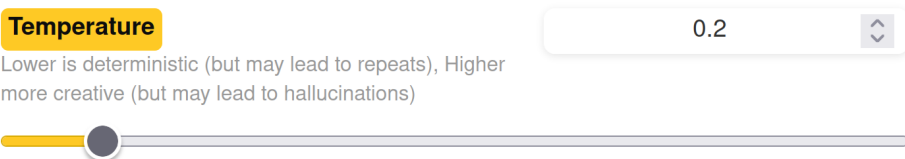
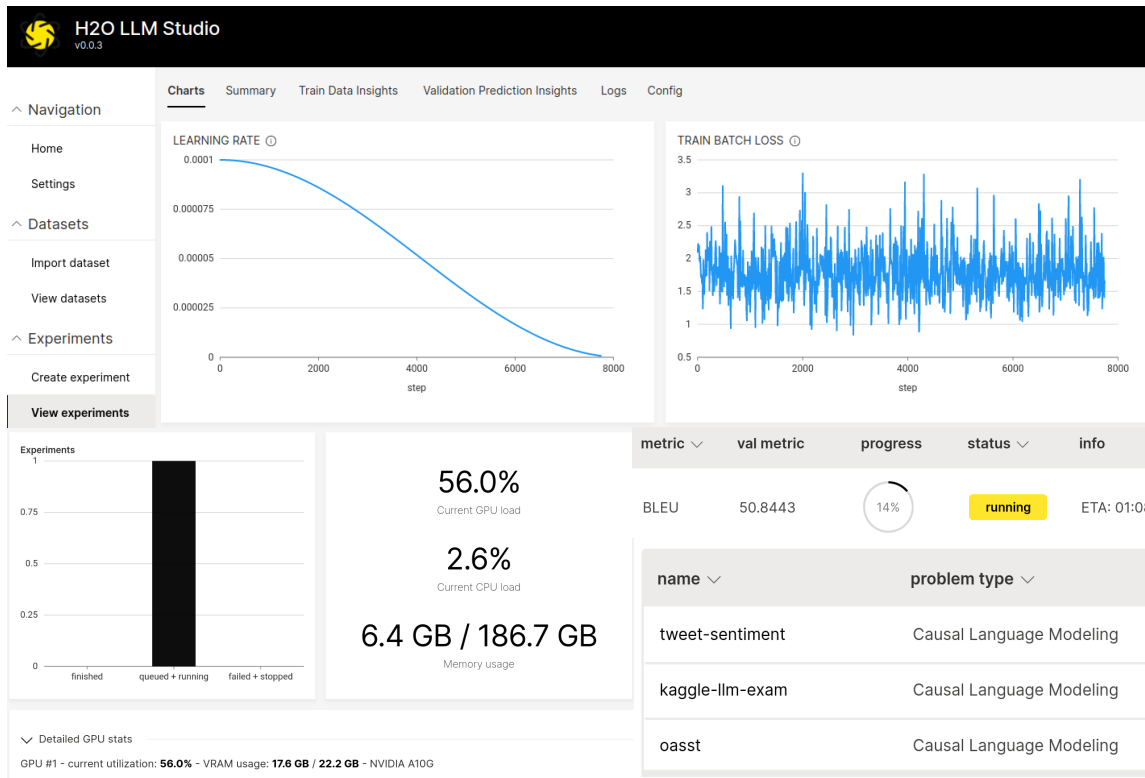


Figure 3: **h2oGPT**. Evaluation of multiple *state-of-the-art* LLM models using the same prompt. This visualization and evaluation allows the user to detect clear differences between the models *e.g.* faster or slower inference, clear hallucinations, common memorized patterns. Demo available at <https://gpt.h2o.ai/> completely free.



**Complete LLM Framework.** Users can track all the experiments and the system's status. The software allows complete *customization* of the experimental setup: dataset and model selection, validation and metrics, optimizer, adapters, RLHF, bit precision, etc.

**Dataset \***

**Problem Type \***

Import config from YAML  Off ⓘ

**Experiment Name**

**LLM Backbone**

**Advanced Settings.** Users can use *state-of-the-art* techniques to speed up training and obtain real-time performance metrics. Also we allow Tokenizer and context customization.

Backbone Dtype

Gradient Checkpointing  On ⓘ

Force Embedding Gradients  Off

Intermediate Dropout

Use RLhf  On ⓘ

Reward Model

Adaptive KI Control  On ⓘ

Initial KI Coefficient

Lora  On ⓘ

Lora R

Lora Alpha

Figure 4: **LLM Studio** allows efficient training and fine-tuning of LLMs using *state-of-the-art* techniques (e.g., advanced models, LoRA, int4, RLHF), and an intuitive GUI with complete experiment's customization. More information in <https://github.com/h2oai/h2o-llmstudio>. Apache 2 License.

## Broad Impact

We advocate for the use of open-source LLMs to accelerate AI development and enhance its transparency, accessibility, security, and reliability. Our open framework for training, fine-tuning, deployment and analysis of LLMs enables this to any user, in a private and safe manner. We provide a detailed [Disclaimer](#) for users of our software.

## References

- Sid 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. *arXiv preprint arXiv:2204.06745*.
- 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](#).
- Arno Candel, Jon McKinney, Philipp Singer, Pascal Pfeiffer, Maximilian Jeblick, Prithvi Prabhu, Jeff Gambera, Mark Landry, Shivam Bansal, Ryan Chesler, et al. 2023. h2ogpt: Democratizing large language models. *arXiv preprint arXiv:2306.08161*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%\\* chatgpt quality](#).
- 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 Pillai, 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](#).
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Xinyang Geng. 2023. [EasyLM: A simple and scalable training framework for large language models](#).
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.
- 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 arXiv:2106.09685*.
- 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*.
- Volodymyr Kuleshov. 2023. Llm-tune: Fine-tuning large language models on one consumer gpu. <https://github.com/kuleshov-group/llmtune>.
- Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Al-lauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2020. [Flaubert: Unsupervised language model pre-training for french](#). In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.
- 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](#).

- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar S. Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.
- Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. [Wizardcoder: Empowering code large language models with evol-instruct](#).
- Mistral AI. 2023. Mistral 7b introduction. <https://mistral.ai/news/announcing-mistral-7b/>.
- MosaicML. 2023. [Mpt-30b: Raising the bar for open-source foundation models](#).
- OpenAI. 2023. [Gpt-4 technical report](#).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 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.
- Tevan Le Scao, Angela Fan, Christopher Akiki, Elie 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*.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca).
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł Ukasz 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.
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, and Nathan Lambert. 2020. Trl: Transformer reinforcement learning. <https://github.com/lvwerra/trl>.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.
- 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.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.
- 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*.
- 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 arXiv:1909.08593*.

# Koala: An Index for Quantifying Overlaps with Pre-training Corpora

Thuy-Trang Vu  Xuanli He  Gholamreza Haffari  Ehsan Shareghi \*

Department of Data Science & AI, Monash University

 Department of Computer Science, University College London

{trang.vu1, gholamreza.haffari, ehsan.shareghi}@monash.edu h.xuanli@ucl.ac.uk

## Abstract

In very recent years more attention has been placed on probing the role of pre-training data in Large Language Models (LLMs) downstream behaviour. Despite the importance, there is no public tool that supports such analysis of pre-training corpora at large scale. To help research in this space, we launch Koala, a searchable index over large pre-training corpora using *lossless* compressed suffix arrays with highly efficient compression rate and search support. In its first release we index the public proportion of OPT 175B, GPT-3, GPT-Neo, GPT-Neo, LLaMA, BERT, ELECTRA, RoBERTa, XLNet pre-training corpora. Koala provides a framework to do forensic analysis on the current and future benchmarks as well as to assess the degree of memorization in the output from the LLMs. Koala is available for public use at <https://koala-index.erc.monash.edu/>.

## 1 Introduction

Large Language Models (LLMs) have achieved state-of-the-art results in NLP and on many benchmarks have reached the performance ceiling (Chowdhery et al., 2022). This evergrowing success has been facilitated by the algorithmic and computational progress in scaling up model sizes (Wei et al., 2022a; Chowdhery et al., 2022; Zhang et al., 2022; Brown et al., 2020), integrating human feedback (Ouyang et al., 2022), adopting modes of instructional inference at both zero- or few-shot settings (Chen et al., 2022; Kojima et al., 2022; Wei et al., 2022b; Nye et al., 2021), as well as the ability of feeding them massive volumes of free text during pre-training.

Recent works exhibit various cases which highlight the sensitivity of downstream behaviour of LLMs (and their smaller variants) to the frequency of observed overlap between pre-training corpora

and test set (Carlini et al., 2022; Tänzler et al., 2022; Razeghi et al., 2022; Magar and Schwartz, 2022; Lewis et al., 2020). In the generative setting, several issues such as hallucination (Dziri et al., 2022), undesired biases (Feng et al., 2023; Kirk et al., 2021), or toxicity (Gehman et al., 2020) have been attributed partly or fully to the characteristics of the pre-training data, while a parallel line of works have emphasised on the positive role of filtering the pre-training data for safety and factual grounding (Thoppilan et al., 2022).

The above observations are not a comprehensive list but echo *the undeniable role of pre-training data in how these models would function in practice*. Understanding the limitations imposed by pre-training data would also lead to more informed algorithmic and computational innovations (Collier et al., 2022). However, these forensic studies are done either at a small scale or by using surrogate sources such as web search hit counts. This is mainly due to the absence of reliable tools for supporting deeper analyses in this space at large scale. Our work attempts to fill this gap.

We launch the Koala project, a service backed by *lossless* compressed suffix arrays (CSA) (Navarro and Mäkinen, 2007), with efficient compression rate and query support. Koala contains a searchable index over the public portion of the pre-training corpora<sup>1</sup> of several existing pre-trained language models from OPT 175B (Zhang et al., 2022) to BERT (Devlin et al., 2019a). Koala is intended to provide various overlap statistics for text query files provided by researchers. We foresee several areas of impact for Koala; (i) as a tool to measure data leakage between existing benchmarks and pre-training corpora of LLMs, (ii) and evaluate the degree of memorisation or creativity in generative models' output, (iii) and to support designing harder benchmarks by reducing the overlap with

\* Corresponding author

<sup>1</sup>Our coverage of pre-training corpora is growing.

pre-training corpora. We present an overview of the `Koala` pipeline for pre-processing and constructing the index. We also provide examples of the types of analyses that could be done via `Koala` by looking at a few commonly used test benchmarks.

## 2 Pre-processing and Corpora Coverage

### 2.1 Pre-processing Steps

Our pre-processing pipeline includes three main steps: cleaning, deduplication and tokenization<sup>2</sup>. The cleaning step varies according to the pre-trained corpus and is described in Section 2.2 where we introduce the corpora covered by `Koala`. In this section, we describe the deduplication and tokenization steps which are shared across all pre-trained corpora.

We use `MinHashLSH` (Rajaraman and Ullman, 2011, Chapter 3)- a widely-adopted duplicate detection method for large-scale dataset, in the deduplication step. Documents are first converted into a set of unigram tokens (shingling) and then are hashed into a short signature, namely minhash, such that the similarity among documents is preserved. `MinHash` is a hashing algorithm based on permutation to generate random hashes to approximate the Jaccard similarity (Broder, 1997; Cohen et al., 2001). We generate the minhashes with 100 permutations. Finally, the locality-sensitive hashes (LSH) of the minhash values are calculated to detect the duplicated candidate pairs. We follow Zhang et al. (2022) to remove those having Jaccard similarity scores above 0.95 threshold. Our deduplication implementation is based on the `datasketch` library.<sup>3</sup> To scale the deduplication process to the large corpus, we first perform deduplication in a small batch and gradually merge the deduplicated batches. *The deduplication, by far, proved to be the most time consuming step of our pre-processing and takes 2-3 orders of magnitude longer than indexing itself. We only applied deduplication to a corpus if the models trained on that corpus also have done so (i.e., according to their corresponding published details).*

The deduplicated corpus is then tokenized with `Moses` (Koehn et al., 2007) to normalize punctuation and remove non-printing characters.

<sup>2</sup>While most existing LLMs use more sophisticated forms of tokenization (i.e., `BytePiece`, `SentencePiece`) we choose `Moses` tokenization as measuring data overlap under token boundaries is a more interpretable and intuitive metric.

<sup>3</sup><https://github.com/ekzhu/datasketch>

### 2.2 Corpora Coverage

The latest version of `koala` at the time of writing this manuscript covers the following corpora:<sup>4</sup>

**BookCorpus** (Zhu et al., 2015) is a large-scale dataset of text derived from books across various genres and topics. We obtained this corpus from `Hugging Face`<sup>5</sup>. This dataset has been used in pretraining multiple large language models such as `BERT` (Devlin et al., 2019b), `RoBERTa` (Liu et al., 2019), `GPT3` (Brown et al., 2020) and `OPT` (Zhang et al., 2022).

**CCNewsv2** contains a vast collection of news articles. Followed Zhang et al. (2022), we extracted English news published between 2016 and 09/2021 from `CommonCrawl` (Nagel, 2016) using `news-please` (Hamborg et al., 2017). Several large language models have utilized this dataset for pretraining purposes, including `RoBERTa` (Liu et al., 2019), `GPT-Neo` (Black et al., 2021) and `OPT` (Zhang et al., 2022).

**ThePile** (Gao et al., 2021) includes datasets from multiple sources: `Pile-CC`, `USPTO Backgrounds`<sup>6</sup>, `Guthenberg` (Rae et al., 2020), `OpenWebTexts` (Gokaslan and Cohen, 2019), `OpenSubtitles` (Tiedemann, 2016), `Wikipedia (en)`, `DM Mathematics` (Saxton et al., 2019), `HackerNews`<sup>7</sup>, `Enron Emails` (Klimt and Yang, 2004), `EuroParl` (Koehn, 2005), `FreeLaw`<sup>8</sup>, `NIH Exporter`<sup>9</sup>, `PhilPapers`<sup>10</sup>, `PubMed Central`, `PubMed Abstracts`, `Stack Exchange`<sup>11</sup>, `Ubuntu IRC`<sup>12</sup> and `YoutubeSubtitles`. Several language models, such as `GPT-Neo` (Black et al., 2021), `OPT` (Zhang et al., 2022) and `LLaMA` (Touvron et al., 2023), have used either all or a portion of the `Pile` dataset as part of their pretraining data.

**Pushshift Reddit** is a project that collects and provides access to `Reddit` data for research and analysis<sup>13</sup>. We used `langdetect`<sup>14</sup> to detect and extract

<sup>4</sup>We plan to index more public pre-training corpora as they become available.

<sup>5</sup><https://huggingface.co/datasets/bookcorpus>

<sup>6</sup><https://bulkdata.uspto.gov>

<sup>7</sup><https://news.ycombinator.com>

<sup>8</sup><https://www.courtlistener.com>

<sup>9</sup><https://exporter.nih.gov>

<sup>10</sup><https://philpapers.org/>

<sup>11</sup><https://archive.org/details/stackexchange>

<sup>12</sup><https://irclogs.ubuntu.com/>

<sup>13</sup><https://files.pushshift.io/reddit>

<sup>14</sup><https://github.com/fedlopez77/langdetect>

| CORPUS           | RAW       | DEDUPLICATION |           | CSA INDEXING |           |
|------------------|-----------|---------------|-----------|--------------|-----------|
|                  | SIZE (GB) | TIME (MIN)    | SIZE (GB) | TIME (MIN)   | SIZE (GB) |
| Enron Emails     | 1.4       | -             | -         | 9.4          | 1.4       |
| NIH ExPorter     | 2         | -             | -         | 21.7         | 1.4       |
| PhilPapers       | 2.5       | -             | -         | 36.6         | 2.5       |
| YoutubeSubtitles | 3.9       | -             | -         | 63.8         | 5.3       |
| HackerNews       | 3.9       | 7,147.2       | 3.2       | 34.2         | 3.3       |
| BookCorpus       | 4.3       | 14,301.2      | 3.7       | 88.1         | 3.6       |
| EuroParl         | 4.7       | -             | -         | 72.3         | 3.7       |
| Ubuntu IRC       | 5.9       | -             | -         | 106.5        | 6.5       |
| DM Mathematics   | 7.8       | 7,881.6       | 1.7       | 32.5         | 3.7       |
| OpenSubtitles    | 13        | 19,920.1      | 4.9       | 58.1         | 4.8       |
| Guthenberg       | 10.9      | 23,893.0      | 9.7       | 139.0        | 9.5       |
| Wikipedi         | 17        | 31,124.4      | 14        | 160.4        | 13        |
| PubMed Abstracts | 20        | -             | -         | 368.5        | 15        |
| USPTO            | 22.9      | 41,866.8      | 22        | 206.8        | 16        |
| Stack Exchange   | 33        | -             | -         | 684.1        | 39        |
| FreeLaw          | 51        | -             | -         | 854.3        | 43        |
| OpenWebTexts     | 62.8      | 115,088.2     | 54        | 885.8        | 47        |
| PubMed Central   | 90        | -             | -         | 2066.7       | 85        |
| Books3           | 104       | -             | -         | 2523.2       | 93        |
| CCNewsv2         | 150       | 292,724.7     | 94        | 818.3        | 80        |
| Pile-CC          | 227.1     | 416,186.8     | 123       | 1,965.2      | 106       |
| Reddit           | 420       | 617,906.5     | 345       | 4,821.2      | 358       |

Table 1: Statistics of corpora, deduplication step, and the index construction. Indexing is done on a single CPU core of a 2.70 GHz Intel Xeon Gold 6150, and requires  $2.5\times$  of index size of RAM memory.

the English comments and submissions posted from 2005 to 2019. We followed pre-processing procedure in (Roller et al., 2021) to remove the post from known non-English subreddits and bot<sup>15</sup>, comments longer than 2048 characters or containing URL, or at depth larger than 7 in a thread. The dataset constitutes a substantial portion of the pretraining data for OPT (Zhang et al., 2022).

Table 1 reports the size of each corpus in raw and deduplicated (if applicable) version.

### 3 Pipeline and Features of Koala

#### 3.1 Data Structure of Koala

Our index construction is inspired by the language models of Shareghi et al. (2015), which leverage compressed data structures for building language models on large text corpora. In this subsection we provide a brief overview of the data structures behind Koala and refer the readers to Shareghi et al. (2016) for further details on the compression framework.

A Suffix Array (SA) (Manber and Myers, 1993) of a string  $\mathcal{T}$  with alphabet  $\sigma$  is an array of its sorted suffixes. A cell in a suffix array, denoted by  $SA[i]$ , stores a number indicating the starting position of its corresponding suffix in  $\mathcal{T}$ . Using

<sup>15</sup><https://github.com/eliassjogreen/Reddit-Bot-List>

a suffix array, searching for any sequence  $\mathbf{u}$  in  $\mathcal{T}$  translates into a binary search to find the range that spans over all substrings that have  $\mathbf{u}$  as their prefix, and is  $\mathcal{O}(|\mathbf{u}| \log |\mathcal{T}|)$ . Constructing SA takes  $4-8|\mathcal{T}|$  bytes in practice, making them impractical to use for large data.

To support search on large collections, Compressed Suffix Array exploits the compressibility of  $\mathcal{T}$  while providing the same functionality of SA in space equal to bzip2 compressed  $\mathcal{T}$  in practice. We follow Shareghi et al. (2016) and use the FM-Index (Ferragina et al., 2008) that utilises the *lossless* text compressibility via the Burrows-Wheeler transformation (BWT) (Burrows and Wheeler, 1994) of the text. The BWT is defined as,  $BWT[i] = [SA[i] - 1 \bmod |\mathcal{T}|]$ . Searching for a sequence in BWT is done in reverse order and requires  $\mathcal{O}(|\mathbf{u}| \log |\sigma|)$ . For more details on BWT and reverse searching, refer to Navarro and Mäkinen (2007).

The CSA is at the core of Koala’s index and search backbone. We used the SDSL library (Gog et al., 2014) to implement our corpus indexer. We index each corpus separately. Once a corpus is indexed, its constructed index sits on disk and could be queried through the Koala web interface (introduced shortly). Each query is launched into the indexed collection of corpora and returns the hit counts of the query in the corresponding corpus. Table 1 reports the time and memory usage for construction of indexes.

#### 3.2 $n$ -gram Overlap Statistics of Koala

Given a text query, Koala can provide its count statistics in several pretraining corpora by querying the indexes constructed. An example of the raw count output for the phrase *plastic bags floating in the ocean* is shown in Table 2 on OPT 175B pretraining corpora. Meaningful insights can be derived from these raw statistics. Figure 1 illustrates two high-level statistics built on top of the  $n$ -gram counts for two question answering benchmark test sets, PIQA (Bisk et al., 2020) and OpenBookQA (Mihaylov et al., 2018), highlighting the amount of leakage or overlap that exists between these test sets and the entire pre-training data collection indexed in Koala. We first introduce how these statistics are calculated per instance, noting that Figure 1 is reporting them as an average across all instances in each test set. The high-level statistics are defined as follows:



| $n$ | $n$ -grams list                    | Pile-CC    | BookCorpus | CCNewsv2  | DM      | Gutenberg | HackerNews | OpenSubtitles | OpenWebTexts | USPTO     | Wikipedia | Reddit     |
|-----|------------------------------------|------------|------------|-----------|---------|-----------|------------|---------------|--------------|-----------|-----------|------------|
| 1   | plastic                            | 959364     | 33845      | 580607    | 0       | 4964      | 14397      | 14114         | 329535       | 598625    | 39435     | 2650049    |
|     | bags                               | 579401     | 29213      | 415672    | 0       | 17160     | 5405       | 21590         | 166685       | 111115    | 13708     | 1697726    |
|     | floating                           | 303836     | 19752      | 162095    | 0       | 36242     | 10058      | 8165          | 120146       | 244489    | 21938     | 976575     |
|     | in                                 | 355723492  | 9260245    | 308475794 | 3347881 | 30592137  | 7135629    | 7831355       | 150523086    | 63002717  | 54190836  | 749899134  |
|     | the                                | 1056004732 | 34886372   | 782874590 | 6519155 | 107380032 | 20809865   | 23296159      | 428544710    | 251429575 | 128120455 | 2128039302 |
|     | ocean                              | 575919     | 30175      | 273507    | 0       | 65172     | 8467       | 23233         | 235331       | 23909     | 41516     | 1125595    |
| 2   | plastic bags                       | 39722      | 845        | 38094     | 0       | 0         | 588        | 367           | 19323        | 7544      | 1267      | 79539      |
|     | bags floating                      | 77         | 4          | 57        | 0       | 0         | 2          | 2             | 25           | 0         | 5         | 275        |
|     | in the                             | 29619      | 3326       | 19189     | 0       | 3492      | 408        | 1397          | 12907        | 2913      | 1695      | 101880     |
|     | ocean                              | 91136626   | 2440752    | 81218136  | 52379   | 7948909   | 1572721    | 1925941       | 37928620     | 19087529  | 13710461  | 175900138  |
|     | plastic bags floating              | 34         | 0          | 22        | 0       | 0         | 1          | 0             | 12           | 0         | 2         | 110        |
|     | in the                             | 27         | 0          | 34        | 0       | 0         | 0          | 0             | 8            | 0         | 3         | 101        |
| 3   | floating in the                    | 14481      | 1621       | 10734     | 0       | 1791      | 141        | 725           | 6594         | 1760      | 897       | 43090      |
|     | ocean                              | 44233      | 1573       | 28680     | 0       | 2025      | 1035       | 2513          | 21517        | 1588      | 2566      | 163343     |
|     | plastic bags floating in           | 16         | 0          | 10        | 0       | 0         | 0          | 0             | 3            | 0         | 2         | 43         |
|     | the                                | 20         | 0          | 29        | 0       | 0         | 0          | 0             | 5            | 0         | 3         | 76         |
|     | ocean                              | 580        | 19         | 413       | 0       | 7         | 10         | 16            | 372          | 24        | 42        | 2078       |
|     | plastic bags floating in the       | 13         | 0          | 8         | 0       | 0         | 0          | 0             | 1            | 0         | 2         | 33         |
| 5   | bags floating in the               | 4          | 0          | 2         | 0       | 0         | 0          | 0             | 1            | 0         | 2         | 9          |
|     | ocean                              | 4          | 0          | 2         | 0       | 0         | 0          | 0             | 1            | 0         | 2         | 9          |
| 6   | plastic bags floating in the ocean | 4          | 0          | 2         | 0       | 0         | 0          | 0             | 1            | 0         | 2         | 9          |

Table 2: The  $n$ -gram hit statistics per corpus for the correct answer (*plastic bags floating in the ocean*) to the query *Which of these situations is an example of pollutants?*, choices : [*plastic bags floating in the ocean*, *mallard ducks floating on a lake*, *cottonwood seeds floating in the air*, *cirrus clouds floating in the sky*]. This is a sample from the OpenBookQA benchmark.

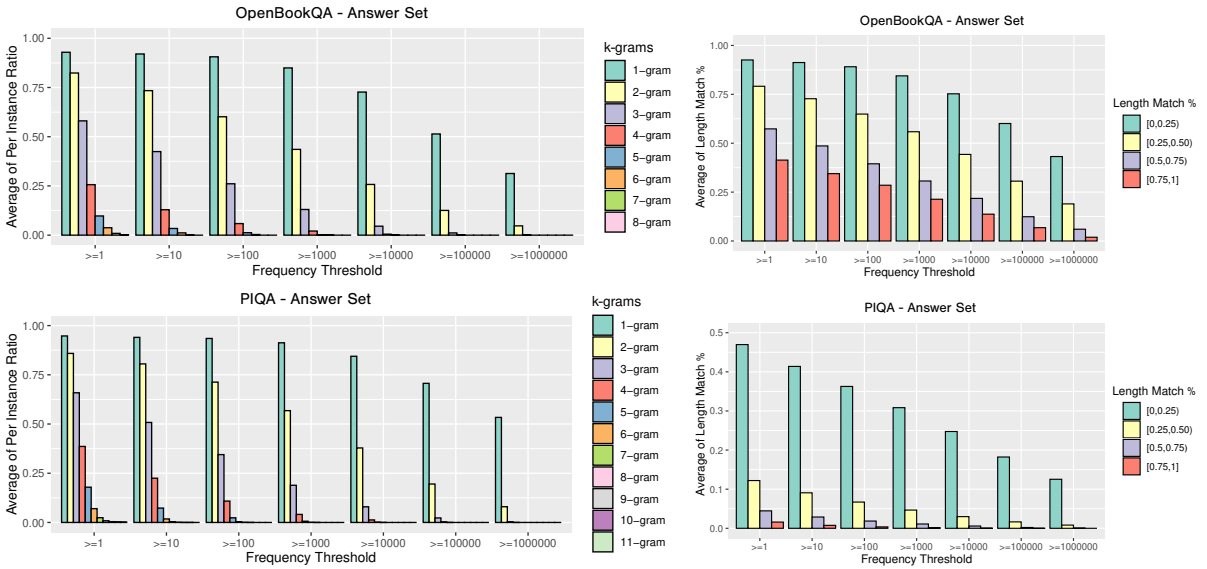


Figure 1: Visualisations of  $n$ -gram overlap statistics for OpenBookQA and PIQA test sets, Answer side. **Top:** OpenBookQA Answer Set ; **Bottom:** PIQA Answer Set. **Left:** Average of Per Instance K-gram hit ratio (i.e., K-gram hit ratio = 1 means 100% of  $k$ -grams in one instance were a hit); **Right:** Average of Per Instance K-gram hit length ratio (i.e., K-gram hit length ratio with respect to the instance length = 1 means the  $k$ -gram was fully covered, 0.75 means it was 3/4 covered, etc). PIQA test set size is 1838, OpenBookQA test set size is 500.

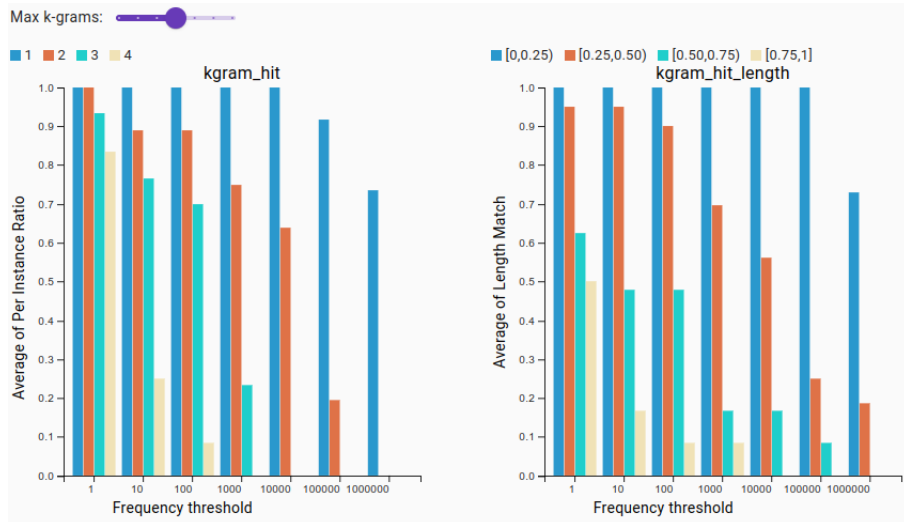
**Per Instance  $k$ -gram hit ratio** measures  $\frac{M_x^{k,t}}{N_x^k}$ , where  $N_x^k$  is the set of all  $k$ -grams of instance  $x$ , and  $M_x^{k,t}$  is the subset of  $N_x^k$  containing only the  $k$ -grams with frequency above the pre-set thresholds  $t$  (e.g.,  $\geq 1$ ,  $\geq 10$ ,  $\geq 100$ ,  $\geq 1k$ ,  $\geq 10k$ ,  $\geq 100k$ ,  $\geq 1M$ ).

**Per Instance  $k$ -gram hit length ratio** measures  $\frac{M_x^{l,t}}{N_x^l}$ , where  $N_x^l$  is the set of all substrings of instance  $x$  that fall within the length bin  $l$  (e.g.,  $l = [0.75, 1.00]$  means all substrings whose lengths are 3/4 of the length of  $x$  or more), and  $M_x^{l,t}$  is the subset of  $N_x^l$ , containing only the substrings with frequency above the pre-set thresholds  $t$  (e.g.,  $\geq 1$ , ...,  $\geq 1M$ ). In this

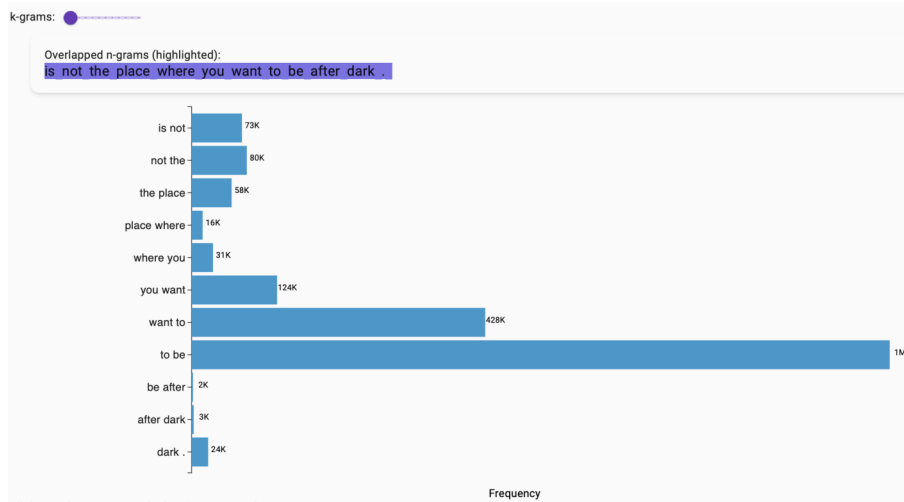
illustration we considered 4 length bins:  $[0, 0.25)$ ,  $[0.25, 0.50)$ ,  $[0.50, 0.75)$ , and  $[0.75, 1]$ .

While a deep dive into exploring the dependence between data overlap, model size, and model performance requires a separate work, here we unpack some highlights from the figures:

**Highlights from Figure 1 (Left Panel):** The top-left panel highlights that for OpenBookQA above 75% of the unigrams and bigrams of test set occur at least once ( $\geq 1$ ) in the pretraining data, while this drops to below 50% with a higher threshold ( $\geq 1k$ ). We observe that above 25% of trigrams occur at least 100 times in the pretraining data. Looking at the bottom-left panel for PIQA, we see a much



(a) Various  $n$ -gram statistics which are available both through the interface and JSON result files.



(b) Count statistics of various  $n$ -grams in the generated text and highlight the overlap  $n$ -grams.

Figure 2: Snapshots from a few of the Koala webpage features.

stronger indication of data overlap. For instance we observe above 55% over bigrams occur at least 100 times in the pre-training data. Comparing the two dataset at the extreme frequency threshold of  $\geq 1M$ , we observe that above 50% of PIQA unigrams occur at least 1M times in the pretraining data, while this is roughly 30% for OpenBookQA.

**Highlights from Figure 1 (Right Panel):** Noting that average answer length in PIQA and OpenBookQA test sets are 101, 20. This means that [0.25,0.5) length bin covers sequences of roughly 25-50 tokens for PIQA, while this is roughly 5-10 tokens for OpenBookQA. We now turn to the highlights from the right panel. For OpenBookQA (top-right) we observe from the red bars that above 25% of test instances (roughly 125 cases out of 500 test instances in OpenBookQA) are almost [75%,100%] covered in the pre-training data for

at least 100 times ( $\geq 100$ ). This corresponds to matches of length 15-20 words. Looking at PIQA (Bottom-Right), although the coverage with respect to the full length is not as apparent as OpenBookQA, matches in each corresponding length bin of PIQA are roughly  $4\times$  longer than OpenBookQA. For instance, about 5% of test instances of PIQA (roughly 90 cases out of 1838 test instances in PIQA) have a matching substring of 25-50 words which occur at least 1000 times in the pretraining data (see yellow bar for  $\geq 1000$ ).

The performance ceiling obtained by GPT-3 and OPT models for these two benchmarks (reported numbers in Appendix A of Zhang et al. (2022) indicate the largest variant of both models achieve roughly 80% accuracy for PIQA, and above 57% accuracy on OpenBookQA) and our highlighted findings suggests a positive correlation between the

amount of data overlap we highlighted and the task performance ceiling by the LLMs trained on the same pre-training corpora. As a future direction of analysis, it would be interesting to leverage `Koala` to analyse the interdependence of the amount of data overlap, model size, and task performance.

### 3.3 Interface of `Koala`

In this section, we give an overview of the interface of `Koala`. Figure 2a and 2b demonstrate some of `Koala`'s features. In addition to reporting the raw counts, `Koala` provides an interface to upload an  $n$ -gram file and to visualize different hit ratio statistics (§3.2). The  $n$ -gram file is a plain text file where each line is an  $n$ -gram whose overlap statistics will be computed. Figure 2a shows the output from this feature. We also provide the interactive version of the ratio plots (e.g., Figure 1) for 3 question answering benchmarks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020) and OpenBookQA (Mihaylov et al., 2018) where overlap and memorization are critical in the evaluation.

For resource management, we limit the live demo queries to  $n$ -gram files below 2MB. For larger files and more comprehensive statistics, we provide a form for users to submit the data and queue the computation. Upon completion (within 72 hours depending on the queuing load), a JSON file is returned to the user with overlap breakdowns per pre-training corpus for various  $n$ -gram lengths. The query files and JSON file are only kept for 72 hours, after which we deep delete them from the server.

Another use case of the overlap statistics is to provide a measure of the creativity for generative LLMs, i.e. whether the generated text is novel or memorization of the pretraining corpora. `Koala` implements a tool to verify the novelty of an output of generative LLM given a prompt. Figure 2b shows an example of this feature which provides the count statistics of the  $n$ -grams in the generated text and highlight the overlap  $n$ -grams.

## 4 Conclusion and Future Work

We presented `Koala`, a web-based service powered by a compressed data structure backbone that facilitates efficient search over large collections of texts. `Koala` is a tool for comprehensive overlap analysis with potential use-cases including but not limited to assessing leakage of test benchmarks, measuring the degree of memorization in genera-

tive LLMs outputs. Additionally, `Koala` not only provides a public tool for forensic analysis of these phenomena it could also help benchmark designers towards constructing more challenging testbeds for LLMs.

## Acknowledgments

The authors are grateful to the anonymous reviewers for their helpful comments. The computational resources of this work are supported by the Multimodal Australian ScienceS Imaging and Visualisation Environment (MASSIVE)<sup>16</sup> and Monash Research Cloud.

## References

- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.
- Sid Black, Leo Gao, 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.
- Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE.
- 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.
- Michael Burrows and David Wheeler. 1994. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation Systems Research Center.
- Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.
- 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

<sup>16</sup>[www.massive.org.au](http://www.massive.org.au)

- language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C. Yang. 2001. **Finding interesting associations without support pruning.** *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78.
- Nigel H. Collier, Fangyu Liu, and Ehsan Shareghi. 2022. **On reality and the limits of language data.** *CoRR*, abs/2208.11981.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. **BERT: pre-training of deep bidirectional transformers for language understanding.** In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. **BERT: Pre-training of deep bidirectional transformers for language understanding.** 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.
- Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. 2022. **On the origin of hallucinations in conversational models: Is it the datasets or the models?** In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5271–5285, Seattle, United States. Association for Computational Linguistics.
- Shangbin Feng, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. 2023. **From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair NLP models.** In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11737–11762, Toronto, Canada. Association for Computational Linguistics.
- Paolo Ferragina, Rodrigo González, Gonzalo Navarro, and Rossano Venturini. 2008. **Compressed text indexes: From theory to practice.** *ACM J. of Exp. Algorithms*, 13.
- 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.** *CoRR*, abs/2101.00027.
- Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. **RealToxicityPrompts: Evaluating neural toxic degeneration in language models.** In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.
- Simon Gog, Timo Beller, Alistair Moffat, and Matthias Petri. 2014. **From theory to practice: Plug and play with succinct data structures.** In *Experimental Algorithms - 13th International Symposium, SEA 2014, Copenhagen, Denmark, June 29 - July 1, 2014. Proceedings*, volume 8504 of *Lecture Notes in Computer Science*, pages 326–337. Springer.
- Aaron Gokaslan and Vanya Cohen. 2019. **OpenWeb-Text corpus.**
- Felix Hamborg, Norman Meuschke, Corinna Breitingner, and Bela Gipp. 2017. **news-please: A generic news crawler and extractor.** In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.
- Hannah Rose Kirk, Yennie Jun, Filippo Volpin, Haider Iqbal, Elias Benussi, Frederic Dreyer, Aleksandar Shtedritski, and Yuki Asano. 2021. **Bias out-of-the-box: An empirical analysis of intersectional occupational biases in popular generative language models.** *Advances in neural information processing systems*, 34:2611–2624.
- Bryan Klimt and Yiming Yang. 2004. **The enron corpus: A new dataset for email classification research.** In *European conference on machine learning*, pages 217–226. Springer.
- Philipp Koehn. 2005. **Europarl: A parallel corpus for statistical machine translation.** In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. **Moses: Open source toolkit for statistical machine translation.** In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.
- 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*.
- Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2020. **Question and answer test-train overlap in open-domain question answering datasets.** *arXiv preprint arXiv:2008.02637*.

- 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*.
- Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. *arXiv preprint arXiv:2203.08242*.
- Udi Manber and Eugene W. Myers. 1993. [Suffix arrays: A new method for on-line string searches](#). *SIAM Journal on Computing*, 22(5):935–948.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. [Can a suit of armor conduct electricity? a new dataset for open book question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- Sebastian Nagel. 2016. [CC-News](#).
- Gonzalo Navarro and Veli Mäkinen. 2007. Compressed full-text indexes. *ACM Comp. Surv.*, 39(1):2.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.
- 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*.
- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. [Compressive transformers for long-range sequence modelling](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Anand Rajaraman and Jeffrey David Ullman. 2011. *Mining of Massive Datasets*. Cambridge University Press, USA.
- Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*.
- Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. [Recipes for building an open-domain chatbot](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.
- David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. [Analysing mathematical reasoning abilities of neural models](#). In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2015. Compact, efficient and unlimited capacity: Language modeling with compressed suffix trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2016. Fast, small and exact: Infinite-order language modelling with compressed suffix trees. *Transactions of the Association for Computational Linguistics*, 4:477–490.
- Michael Tänzler, Sebastian Ruder, and Marek Rei. 2022. Memorisation versus generalisation in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7578.
- Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. [Lamda: Language models for dialog applications](#). *arXiv preprint arXiv:2201.08239*.
- Jörg Tiedemann. 2016. [Finding alternative translations in a large corpus of movie subtitle](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).
- 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*.
- 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 Chi, Quoc Le, and Denny Zhou. 2022b. [Chain of thought prompting elicits reasoning in large language models](#). *arXiv preprint arXiv:2201.11903*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

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*.

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 *The IEEE International Conference on Computer Vision (ICCV)*.

# Sudowoodo: a Chinese Lyric Imitation System with Source Lyrics

Yongzhu Chang<sup>1</sup>, Rongsheng Zhang<sup>1,2\*</sup>, Lin Jiang<sup>3</sup>, Qihang Chen<sup>3</sup>  
Le Zhang<sup>1</sup>, Jiashu Pu<sup>1</sup>

<sup>1</sup> Fuxi AI Lab, NetEase Inc., Hangzhou, China

<sup>2</sup> Zhejiang University

<sup>3</sup> Music AV Lab, NetEase Inc., Hangzhou, China

{changyongzhu, zhangrongsheng}@corp.netease.com

## Abstract

Lyrics generation is a well-known application in natural language generation research, with several previous studies focusing on generating accurate lyrics using precise control such as keywords, rhymes, etc. However, lyrics imitation, which involves writing new lyrics by imitating the style and content of the source lyrics, remains a challenging task due to the lack of a parallel corpus. In this paper, we introduce *Sudowoodo*, a Chinese lyrics imitation system that can generate new lyrics based on the text of source lyrics. To address the issue of lacking a parallel training corpus for lyrics imitation, we propose a novel framework to construct a parallel corpus based on a keyword-based lyrics model from source lyrics. Then the pairs (*new lyrics*, *source lyrics*) are used to train the lyrics imitation model. During the inference process, we utilize a post-processing module to filter and rank the generated lyrics, selecting the highest-quality ones. We incorporated audio information and aligned the lyrics with the audio to form the songs as a bonus. The human evaluation results show that our framework can perform better lyric imitation. Meanwhile, the *Sudowoodo* system and demo video of the system is available at *Sudowoodo* and [https://youtu.be/u5BBT\\_j1L5M](https://youtu.be/u5BBT_j1L5M).

## 1 Introduction

AI creative assistants are artificial intelligence systems that can learn from large amounts of text data to understand human language and culture and use this knowledge to create content such as story generation (Alabdulkarim et al., 2021; Zhu et al., 2020), poetry writing (Guo et al., 2019; Liu et al., 2019b; Yang et al., 2019), grammar and spelling checking (Patil et al., 2021), etc. In addition, AI creative assistants can also assist in songwriting (Potash et al., 2015; Zhang et al., 2020; Shen et al., 2019) by learning from numerous songs, understanding human emotional expression, and creating

music in a similar writing style to humans. Previous research (Castro and Attarian, 2018; Watanabe et al., 2018; Manjavacas et al., 2019; Fan et al., 2019; Li et al., 2020; Zhang et al., 2020, 2022) has focused on generating lyrics based on specified keywords (e.g., *Snow*), lyrics styles, themes, or user input passages, which generate new lyrics with limited control over the content. However, in actual music production, users sometimes adapt excellent songs by adding their own creativity while remaining the original lyrical structure, resulting in new lyrics. This requires stronger control over the source lyrics such as text content, emotion, and fine-grained writing styles.

To address this issue, this paper demonstrates *Sudowoodo*<sup>1</sup> (a Pokémon with the ability to imitate) a Chinese lyrics imitation generation system based on source lyrics. *Sudowoodo* is typically based on the Encoder-Decoder framework, where the encoder encodes the text and attributes of the source lyrics, and the decoder generates the imitated lyrics. However, since we only have the source lyrics and not the target ones, the parallel corpus is lacking to train the imitation model. To solve the problem, we also propose a method for constructing aligned training samples, which generated the target lyrics from the extracted keywords of source lyrics using a keywords-based lyrics generation model.

Specifically, we first collect the source lyrics corpus  $D_k$  from the Internet<sup>2</sup> and utilize the keyword extraction method described in Section 2.1 to extract keywords from source lyrics. And we train a keywords-based model, named  $\text{Model}_{K2L}$ , which can generate lyrics from given keywords. Then, we generate the target lyrics  $D_{k'}$  using the  $\text{Model}_{K2L}$ . Finally, we train a lyrics imitation model with the aligned lyrics corpus ( $D_{k'}$ ,  $D_k$ ) based on the encoder-decoder framework. In addition, to improve the quality of generated lyrics

\* Corresponding Author

<sup>1</sup><https://en.wikipedia.org/wiki/Talk%3ASudowoodo>

<sup>2</sup><https://music.163.com/>

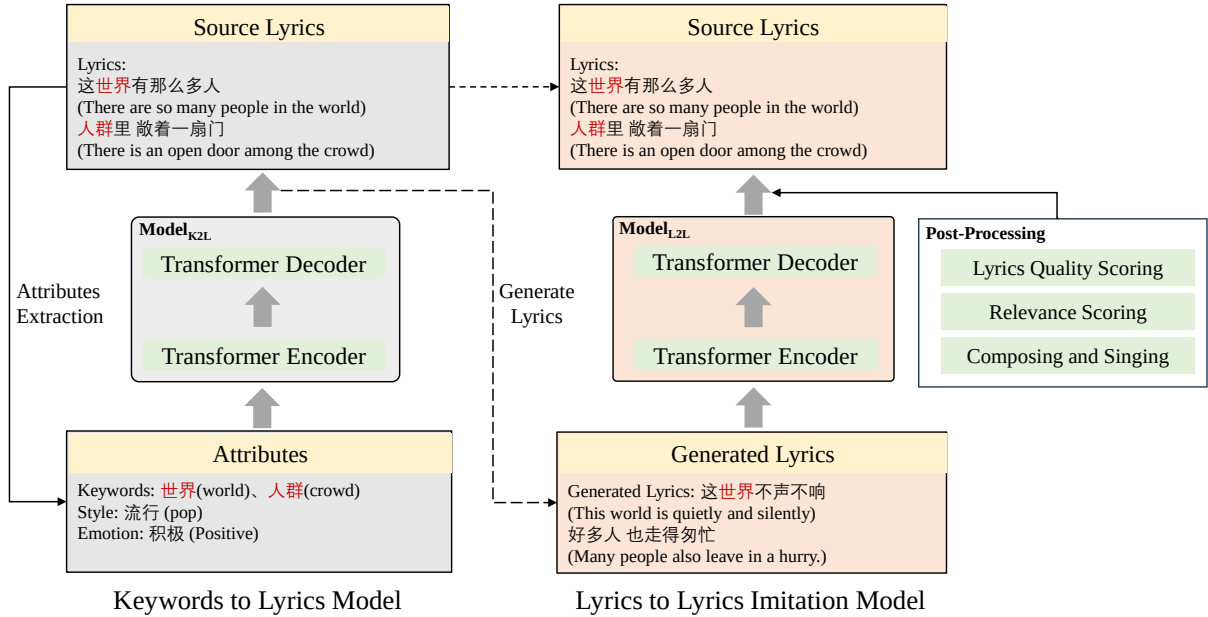


Figure 1: The framework of *Sudowoodo* system proposed in this paper.  $\mathbf{Model}_{K2L}$  denotes a model for generating lyrics based on keywords, while  $\mathbf{Model}_{L2L}$  represents the generation model from source lyrics to imitation lyrics. **Encoder** refers to the encoding portion of the Encoder-decoder architecture, while **Decoder** represents the decoding portion. **Post-processing** is mainly aimed at the imitation lyrics generated based on the  $\mathbf{Model}_{L2L}$ .

and better showcase the results, we also employ post-processing modules including lyrics quality scoring and relevance scoring. Meanwhile, to provide a more intuitive understanding of the generated lyrics through imitation, we incorporate audio information (the vocals and melody of the source song) and align the lyrics with the audio to produce a complete song.

The main contributions of the *Sudowoodo* system are summarized as follows:

- We present a lyric imitation tool that generates new lyrics end-to-end based on source lyrics. Furthermore, we explore the addition of musical information to the generated lyrics in order to create songs. Sample songs can be heard at the songs of [Sudowoodo](#).
- We propose a novel framework for constructing a parallel lyrics corpus for imitation based on the keyword-based model. The results of the human evaluation show the efficacy of the imitation model trained on the basis of this parallel lyrics corpus.
- The *Sudowoodo* system and demo video can be available at [Sudowoodo](#) and [https://youtu.be/u5BBT\\_j1L5M](https://youtu.be/u5BBT_j1L5M).

## 2 Framework

The *Sudowoodo* system consists of two models and a post-processing module, as illustrated in Figure 1:  $\mathbf{Model}_{K2L}$ ,  $\mathbf{Model}_{L2L}$ , and **Post-Processing**. These modules will be described in greater detail below.

### 2.1 Data Preparation

In this study, we obtain a dataset of 800k Chinese lyrics of various styles from the Internet, including pop, hip-hop, rap, etc. After filtering out lyrics less than 100 characters in length and removing duplicates, we are left with 600k unique lyrics. We denote the processed lyrics corpus as  $D_k$ .

As depicted in the attribute extraction section of Figure 1, when conducting attributes extraction for the source lyrics, we extract not only the keywords of the source lyrics but other attributes such as style and emotion. To extract keywords from the source lyrics, we first segment the lyrics into multiple bars. We then apply KBERT (Liu et al., 2019a) based on distiluse-base-multilingual-cased-v1<sup>3</sup> to extract a subset of the keywords from each bar. We extract 5 keywords for each bar. In addition, we rank the

<sup>3</sup>Multilingual knowledge distilled version of multilingual Universal Sentence Encoder. Supports 15 languages: Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, and Turkish. [https://www.sbert.net/docs/pretrained\\_models.html](https://www.sbert.net/docs/pretrained_models.html)



keywords according to their scores and select the top 10% scoring keywords as the keywords for the whole song. In this process, we utilize the Jieba<sup>4</sup> as a word separation tool. For other information, we train a classifier model to acquire attributes such as emotion and style from source lyrics. Finally, we construct a parallel corpus dataset by extracting keywords, style, and emotion from the lyrics and aligning these attributes with source lyrics to form paired data  $(D_A, D_K)$ , where  $D_A$  represents the corpus composed of the extracted attributes of the corresponding source lyrics. The size of this dataset is 600k.

## 2.2 Models

We first train a model, named  $\text{Model}_{K2L}$ , using the paired data  $(D_A, D_K)$  to generate lyrics based on keywords and their associated attributes such as emotion and style. Then, we acquire three new lyrics through  $\text{Model}_{K2L}$  for each source lyric with random keywords extracted from the source lyric. The new lyrics are aligned with the source lyric and keywords to form paired data. All the lyrics generated by  $\text{Model}_{K2L}$  are collected as  $D'_K$ . Consequently, we construct a parallel corpus dataset  $(D'_k, D_k)$  with a size of 1800k. Meanwhile, during the training of  $\text{Model}_{L2L}$ , we encode  $D'_k$  and the write styles of  $D_k$ , while the decoding side targets  $D_k$ .

**Initialization:** To improve the model’s performance and generate more fluent text, we initialize the model with a self-developed transformers-based pre-training model. Note that the structure of the pre-trained model is consistent with GPT-2<sup>5</sup>, containing 210 million parameters with 16 layers, 1024 hidden dimensions, and 16 self-attention heads. The model is pre-trained on 30G of Chinese novels collected from the internet, using a vocabulary of 11400 words and a maximum sequence length of 512.

**Training:** Due to the lack of direct alignment corpus from lyrics to lyrics, we cannot train a seq2seq encoding and decoding model directly. Therefore, we propose a novel training strategy, as shown in Figure 1. The framework comprises two models for training. Firstly, a keyword-to-lyrics model, named  $\text{Model}_{K2L}$ , is used to generate aligned lyrics from source lyrics, with keywords and attributes such as style and emotion encoded

into a latent semantic space and then decoded into source lyrics. The  $\text{Model}_{K2L}$  utilizes an encoder-decoder architecture with the keywords, style, and emotion serving as encoder inputs and the source lyrics as decoder outputs, with training loss as shown in Equation 1. Secondly, an end-to-end lyrics imitation model, called  $\text{Model}_{L2L}$ , is trained using the aligned corpus  $(D'_k, D_k)$  constructed from  $\text{Model}_{K2L}$  and also utilizes the encoder-decoder architecture. The  $\text{Model}_{L2L}$  encodes  $D'_k$  and the attributes of the source lyrics into the encoder, with the source lyrics serving as the decoder output and training loss as shown in Equation 2.

$$L_{K2L} = - \sum_{D_k} \log P(y_i | D(E(k_i, W_i))) \quad (1)$$

$$L_{L2L} = - \sum_{(D'_k, D_k)} \log P(y_i | D(E(x_i, k_i, W_i))) \quad (2)$$

Where  $E$  encodes lyrics, keywords, and writing styles into latent representation, and  $D$  decodes the latent representation into lyrics.  $k_i$  means the keywords and the  $W_i$  represents the writing styles such as emotion and style in source lyrics.  $x_i$  indicates the lyrics in  $D'_k$ .  $D_k$  is the dataset of source lyrics, and  $D'_k$  is lyrics generated from  $\text{Model}_{K2L}$ .

**Inference:** During inference, the input to  $\text{Model}_{K2L}$  is controlled by keywords and writing style and is typically less than 512 in length. In contrast,  $\text{Model}_{L2L}$ ’s inputs include the source lyrics, which can easily exceed the length of 512. The most intuitive approach is to truncate the inputs after incorporating the keywords and writing style. However, this approach would be easy to obscure the controlling elements such as writing style and keywords. To address this issue, when the lyrics exceed 512 minus the length of the writing style and keywords, we truncate the last bar of the source lyrics to ensure that the input to the model does not exceed 512. It is worth noting that the last bar of the source lyrics often repeats the previous content, so this truncation does not significantly impact the generated lyrics.

**Decoding Strategy:** We use a top-k sampling strategy with a sampling temperature of 0.8 and a value of  $k$  of 10. Additionally, to prevent the model from easily generating duplicate words, we apply a sampling penalty technique proposed by Yadong et al. (2021), which only penalizes the first 200 words. In lyrics generation, although the model

<sup>4</sup><https://github.com/fxsjy/jieba>

<sup>5</sup><https://openai.com/blog/gpt-2-1-5b-release/>

can learn the specific format, which is the number of lines and a number of words per line based on the source lyrics, we perform format control decoding to ensure that the generated lyrics have the same format as the source lyrics. To do this, we record the number of lines and words in the generated lyrics and adjust the  $[SEP]$  and  $[EOS]$  logits in each decoding step.

### 2.3 Post-processing

After the model training is finished, we can use the source lyrics, provided keywords, and writing styles to generate limitation lyrics with  $\text{Model}_{L2L}$ . We utilize the top-k sampling method at decoding to generate candidate lyrics. For each input, the model generates 10 samples. Then we re-rank the samples according to the following scores.

**Lyrics Quality Scoring:** To filter high-quality lyrics, we train a classification model to determine whether a song lyric is a high-quality lyric and consider its confidence score as the Lyrics score, which is called  $S_{Lyric}$ , for re-rank. Inspired by QiuNiu (Zhang et al., 2022), we utilize popular and classic lyrics as positive samples, while lyrics with very few plays are negative samples. The experimental results indicate that the model gives a high confidence score when the lyrics contain beautiful sentences and rhetorical devices.

**Relevance Scoring:** In this paper, we introduce a method called  $S_{relevance}$  to measure the semantic similarity between source lyrics and generated lyrics. To calculate  $S_{relevance}$ , we use the sentence transformer to obtain sentence vectors for both the source and generated lyrics, and then calculate the cosine similarity to rank the relevance. This method allows us to evaluate the quality of the generated lyrics in terms of their semantic similarity to the original lyrics.

Finally, we apply an anti-spam filter to the lyrics and use a combination of scores to sort them as shown in Equation 3. We then select the top 3 results as the final output. This post-process allows us to identify the most high-quality lyrics according to our criteria.

$$Score = w_1 * S_{Lyric} + w_2 * S_{relevance} \quad (3)$$

which the  $w_1$  and  $w_2$  denote the weights of the corresponding scores. In this paper, we set  $w_1$  to 0.7 and  $w_2$  to 0.3.

**Composing and Singing:** In order to evaluate the quality of lyrics generated from the  $\text{Model}_{L2L}$ ,

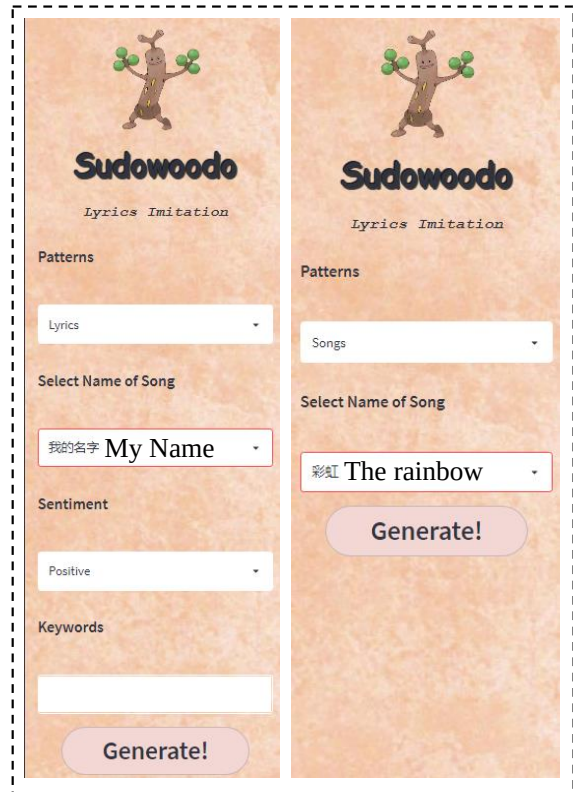


Figure 2: The interface of Sudowoodo.

we annotate popular songs by extracting various musical features, including melody, chord progression, key, structure, and phrasing, using both general music theory<sup>6</sup> and more advanced analytical techniques. Based on these features, we then use intelligent composition (Song et al., 2009) techniques to generate melodies similar to those in the source style. Additionally, we use matching arrangement techniques, virtual vocal timbre selection, and mixing parameter adjustment to produce a fully synthesized song that includes accompaniment and singing. Finally, we incorporated audio information and aligned the lyrics with the audio to form the songs as shown in Figure 2. We can enjoy it in songs mode of Sudowoodo!

### 3 Results of the Experiment

We conduct an ablation study to evaluate the framework proposed in this paper.

**Metrics:** We evaluate the generated lyrics from four perspectives: (1) *Thematic*: The relevance of the imitation lyrics to the theme of the source lyrics, including love, friendship, family inspiration, etc. (2) *Fluency*: It refers to the smoothness

<sup>6</sup><https://www.ipr.edu/blogs/audio-production/what-are-the-basics-of-music-theory/>

|                      | Theme (avg.) | Flu (avg.)   | Logic (avg.) | Overall (avg.) | Best (%)     |
|----------------------|--------------|--------------|--------------|----------------|--------------|
| Model <sub>K2L</sub> | 4.168        | 4.103        | <b>3.480</b> | 4.078          | 32.75        |
| Model <sub>L2L</sub> | 4.250        | <b>4.160</b> | 3.460        | <b>4.153</b>   | <b>34.25</b> |
| w/o WS               | <b>4.275</b> | 4.108        | 3.415        | 4.148          | 33           |

Table 1: Human evaluation results of Ablation. The scores in the table are the average scores of the three annotators. "Best" indicates that the model achieves Top-1 in the validation dataset for the same source lyric using three end-to-end lyrics imitation methods. *Flu* means *Fluency*, and *Theme* is *Thematic* in metrics. WS means the writing styles such as keywords, style, and emotion.



Figure 3: The instance of imitation lyrics in Lyrics mode. We enter the "爱情 (love)" and "自由 (freedom)" as keywords. As you can see from the picture, not all of the keywords entered are necessarily used. The Red color in Chinese and English indicates keywords.

and naturalness of the language used in the lyrics. In evaluating the fluency of a song's lyrics, we consider factors such as the fluency of the words and the rhythmic structure of the sentences. (3) *Logic*: It refers to the coherence and smoothness of scene transitions in the lyrics. To evaluate the logic of a song's lyrics, we consider whether consecutive sentences describe a single scene. If  $m$  consecutive sentences describe a scene, we argue that those sentences are reasonable within logic. If  $n$  consecutive groups of  $m$  sentences are found to exist within  $n$  different scenes, the lyrics are considered to have a high degree of smooth scene transitions overall. The number of scene jumps<sup>7</sup> can measure the logic of the song. (4) *Overall*: The overall scoring of a song's lyrics.

**Results:** We sample 100 lyrics from the source dataset and generate three imitation lyrics for each source lyric. We invite 3 professional lyricists to score each of the 300 lyrics based on *Thematic*, *Fluency*, *Logic*, and *Overall*. The score ranges from

<sup>7</sup>Scene jumps occur when consecutive sentences describe different things or switch abruptly between different sensory perspectives, resulting in an unnatural or jarring transition.

1-5, with 5 being the best and 1 being the worst. The results are shown in Table 1, where all scores are averages for one song. We observe that the thematic and comprehensive scores of Model<sub>L2L</sub> exceeded those of Model<sub>K2L</sub>. Additionally, we also verify the effect of the model, which uses only lyrics as input without keywords and writing style, and find that the addition of keywords improves the fluency of the generated lyrics. When the model is used to generate lyrics for the same lyrics using all three end-to-end methods, we observe that the method based on generated lyrics outperforms the keywords in 67.25% of cases. It indicates that generated lyrics for training can improve the performance of a lyric imitation model.

## 4 Demonstration

This section demonstrates how the *Sudowoodo* system works.

The user interface for this demo is shown in Figure 2. As an imitation demo, it offers limited interaction with the user. The *Sudowoodo* system operates in two modes: *Lyrics* and *Songs*. In *Lyrics* mode, the user is required to select the source



Figure 4: An example of Songs mode, with a player that plays the rendered song with imitation lyrics above the generated lyrics.

lyrics and the desired sentiment for the generated lyrics. Additionally, the user may provide keywords, which are typically space-separated phrases such as "自由 爱情 (*freedom love*)" or, alternatively, left blank. When generating lyrics, the *Sudowoodo* system takes into account the writing style of the selected source lyrics, including its theme, rhymes, and provided keywords, as well as the desired sentiment. The provided keywords are highlighted for easy identification. Note that not all provided keywords are necessarily used in the generated lyrics. In *Songs* mode, the user can select the name of the source lyrics to hear the generated lyrics as a song. Due to technical limitations, the lyrics are rendered offline. In this paper, we apply three different AI singers to provide the sounds. Finally, the user can click "Generate!" to produce the output.

Next, we show some generated examples in Figure 3.

**Lyrics:** The leftmost column of the display lyrics represents the source lyrics selected by the user, while the three columns on the right show the generated imitation lyrics. If the user has entered keywords, these will be highlighted in red within the generated lyrics. This demo can generate smooth, high-quality lyrics in a format and writing style similar to the source lyrics for each generation.

**Songs:** Figure 4 shows the results in Songs mode. As the real-time rendering of songs is a challenging task, we have performed offline render-

ing for this demo. A player is provided above the generated lyrics, which can be clicked on to hear the resulting song after rendering with the imitation lyrics. In the future, we aim to integrate real-time rendering of songs to create a true lyric imitation system that can take source lyrics and generate corresponding songs. More experiences are available in *Sudowoodo*.

## 5 Conclusion

In this paper, we describe *Sudowoodo*, a Chinese lyric imitation system that supports two modes: Lyrics and Songs. In Lyrics mode, users can input keywords to generate imitated lyrics based on existing lyrics. In Songs mode, *Sudowoodo* uses an unspecified technology to generate music that accompanies the imitated lyrics to create a complete song. To address the lack of a lyric-to-lyric alignment corpus, we propose a novel training framework structure to construct a parallel corpus for lyric imitation. Additionally, we apply Chinese pre-trained GPT-2 for initialization. To improve the quality of the generated lyrics, we employ a post-processing module to sort the generated results and select the highest quality ones. Finally, We audio-aligned some of the imitation lyrics to form songs!

## Acknowledgements

This work is supported by the Key Research and Development Program of Zhejiang Province (No.

2022C01011). We would like to thank the anonymous reviewers for their excellent feedback. We are very grateful for the professional markers provided by [NetEase Crowdsourcing](#)<sup>8</sup>.

## References

- Amal Alabdulkarim, Siyan Li, and Xiangyu Peng. 2021. Automatic story generation: Challenges and attempts. *ArXiv*, abs/2102.12634.
- Pablo Samuel Castro and Maria Attarian. 2018. Combining learned lyrical structures and vocabulary for improved lyric generation. *ArXiv*, abs/1811.04651.
- Haoshen Fan, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A hierarchical attention based seq2seq model for chinese lyrics generation. *ArXiv*, abs/1906.06481.
- Zhipeng Guo, Xiaoyuan Yi, Maosong Sun, Wenhao Li, Cheng Yang, Jian na Liang, Huimin Chen, Yuhui Zhang, and Ruoyu Li. 2019. Jiuge: A human-machine collaborative chinese classical poetry generation system. In *Annual Meeting of the Association for Computational Linguistics*.
- Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Rigid formats controlled text generation. *ArXiv*, abs/2004.08022.
- Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2019a. K-bert: Enabling language representation with knowledge graph. In *AAAI Conference on Artificial Intelligence*.
- Yusen Liu, Dayiheng Liu, and Jiancheng Lv. 2019b. Deep poetry: A chinese classical poetry generation system. In *AAAI*.
- Enrique Manjavacas, Mike Kestemont, and Folgert Karsdorp. 2019. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. In *INLG*.
- Kavita. T. Patil, R. P. Bhavsar, and B. V. Pawar. 2021. Spelling checking and error corrector system for marathi language text using minimum edit distance algorithm.
- Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Conference on Empirical Methods in Natural Language Processing*.
- Liangbin Shen, Pei-Lun Tai, Chao-Chung Wu, and Shou-De Lin. 2019. Controlling sequence-to-sequence models - a demonstration on neural-based acrostic generator. In *Conference on Empirical Methods in Natural Language Processing*.
- Xudong Song, Wanchun Dou, and Wei Song. 2009. A workflow framework for intelligent service composition. *2009 Workshops at the Grid and Pervasive Computing Conference*, pages 11–18.
- Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. In *North American Chapter of the Association for Computational Linguistics*.
- Xing Yadong, Xiao-Xi Mao, Li Le, Lin Lei, Chen Yanjiang, Shuhan Yang, Chen Xuhan, Kailun Tao, Li Zhi, Gongzheng Li, Jiang Lin, Liu Siyan, Zhao Zeng, Minlie Huang, Changjie Fan, and Hu Zhipeng. 2021. Kuileixi: a chinese open-ended text adventure game. In *Annual Meeting of the Association for Computational Linguistics*.
- Zhichao Yang, Pengshan Cai, Yansong Feng, Fei Li, Weijiang Feng, Elena Suet-Ying Chiu, and Hong Yu. 2019. Generating classical chinese poems from vernacular chinese. *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, 2019:6155–6164.
- Le Zhang, Rongsheng Zhang, Xiao-Xi Mao, and Yongzhu Chang. 2022. Qiuniu: A chinese lyrics generation system with passage-level input. In *ACL*.
- Rongsheng Zhang, Xiao-Xi 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 *Conference on Empirical Methods in Natural Language Processing*.
- Yutao Zhu, Ruihua Song, Zhicheng Dou, Jianyun Nie, and Jin Zhou. 2020. Scriptwriter: Narrative-guided script generation. In *Annual Meeting of the Association for Computational Linguistics*.

<sup>8</sup><https://fuxi.163.com/productDetail/17>

# ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format

Qi Zhu<sup>\*1</sup> Christian Geishauer<sup>\*2</sup> Hsien-chin Lin<sup>\*2</sup> Carel van Niekerk<sup>\*2</sup>  
Baolin Peng<sup>3</sup> Zheng Zhang<sup>1</sup> Shutong Feng<sup>2</sup> Michael Heck<sup>2</sup> Nurul Lubis<sup>2</sup>  
Dazhen Wan<sup>1</sup> Xiaochen Zhu<sup>4</sup> Jianfeng Gao<sup>3</sup> Milica Gašić<sup>†2</sup> Minlie Huang<sup>†1</sup>

<sup>1</sup>Tsinghua University, Beijing, China

<sup>2</sup>Heinrich Heine University Düsseldorf, Düsseldorf, Germany

<sup>3</sup>Microsoft Research, Redmond, USA

<sup>4</sup>University of Cambridge, Cambridge, England

<sup>1</sup>{zhu-q18,wandz19}@mails.tsinghua.edu.cn {z-zhang,aihuang}@tsinghua.edu.cn

<sup>2</sup>{geishaus,linh,niekerk,heckmi,lubis,gasic}@hhu.de

<sup>3</sup>{bapeng,jfgao}@microsoft.com <sup>4</sup>xz479@cam.ac.uk

## Abstract

Task-oriented dialogue (TOD) systems function as digital assistants, guiding users through various tasks such as booking flights or finding restaurants. Existing toolkits for building TOD systems often fall short of in delivering comprehensive arrays of data, models, and experimental environments with a user-friendly experience. We introduce ConvLab-3: a multifaceted dialogue system toolkit crafted to bridge this gap. Our unified data format simplifies the integration of diverse datasets and models, significantly reducing complexity and cost for studying generalization and transfer. Enhanced with robust reinforcement learning (RL) tools, featuring a streamlined training process, in-depth evaluation tools, and a selection of user simulators, ConvLab-3 supports the rapid development and evaluation of robust dialogue policies. Through an extensive study, we demonstrate the efficacy of transfer learning and RL and showcase that ConvLab-3 is not only a powerful tool for seasoned researchers but also an accessible platform for newcomers<sup>1</sup>.

## 1 Introduction

Task-oriented dialogue (TOD) systems converse with their users in natural language to help them fulfil a task, such as booking a flight or finding a restaurant. Unlike chit-chat dialogues, a critical aspect of these systems is that they are grounded in an ontology that contains domains, slots, and values which describe the dialogue task, i.e. user goal, as well as including domain-specific databases.

<sup>1</sup>ConvLab-3 is publicly available at <https://github.com/ConvLab/ConvLab-3> under Apache License 2.0. The demonstrative video accompanying this paper is available at <https://youtu.be/t6HVTJCeGLo>.

<sup>\*</sup>These authors contributed equally to this work.

<sup>†</sup>These authors share the senior authorship of this work.

There are two distinct capabilities that TOD systems need to exhibit. They need to *track* the state of the dialogue and based on that *decide* on the next action to take in order to steer the conversation towards fulfilling the user’s goal (Young et al., 2007). The architecture of TOD systems typically adopts a modular approach, often encompassing components like dialogue state trackers and policies, and may include language understanding or generation units, as depicted in Figure 1. The complexity of a TOD system necessitates a toolkit with advanced, easily integrable modules allowing for straightforward training, evaluation, and combination.

The vast amount of possible user behaviours and tasks that a TOD system might assist with necessitates the study of generalization and transfer towards new users and datasets. While many datasets for studying task-oriented dialogue have been proposed (Wen et al., 2016; Mrkšić et al., 2017; Byrne et al., 2019; Eric et al., 2020; Rastogi et al., 2020; Zhu et al., 2020a; Feng et al., 2022), the various dialogue, ontology and database formats hinder researchers from validating their models on unseen data. In this work we propose a unified format to bridge the gap between different TOD datasets and models and provide a unified training and evaluation framework that accelerates the study of generalization capabilities. Once a dataset is transformed into the unified format, it can be immediately used by supported models. Similarly, once a model supports the unified format, it can access all supported datasets. This feature reduces the cost of adapting  $M$  models to  $N$  datasets from  $M \times N$  to  $M + N$ .

The dialogue policy, as the decision-making component of a TOD system, is pivotal to the success or failure of a dialogue task. It is typically optimized using reinforcement learning (RL), necessitating additional components such as algorithms,

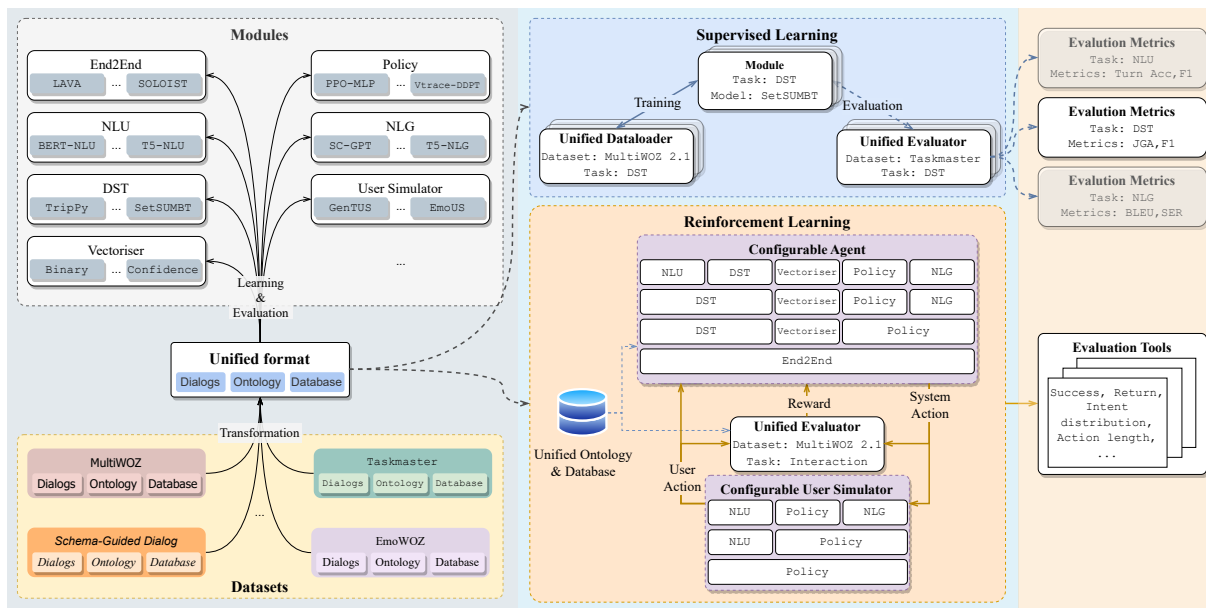


Figure 1: ConvLab-3: The unified format serves as a bridge, connecting diverse datasets and dialogue models. It streamlines the integration of various TOD modules, including supervised learning, evaluation, and a wide array of essential evaluation metrics, thanks to the unified data loader and evaluator. These modules can be incorporated, either in the agent or user simulator, through a configuration file, defining the environment for interactive evaluation and reinforcement learning.

evaluation tools, and user simulators. Realistic user simulators are essential for conducting interactive evaluations and tests against varied user behaviours, in order to accurately mirror real-world scenarios. ConvLab-3 streamlines RL-based development and assessment of dialogue policies. We achieve this by offering a configurable RL environment, evaluation tools for thorough insights, and multiple user simulators to explore generalization capabilities towards new user behaviours, as depicted in Figure 1.

ConvLab-3 is especially useful for practitioners seeking to construct a dialogue system without extensive expertise. Additionally, it provides a fast, convenient, and dependable platform for both novice and experienced researchers to conduct experiments. In particular, it enables: (1) researchers to perform experiments across a variety of datasets, (2) developers to construct an dialogue system using custom datasets, and (3) community contributors to consistently add models and datasets. In summary, our contributions are:

- A unified data format which allows for easy generalisation and transfer learning experiments across different datasets.
- A convenient RL framework and access to different user simulators, accelerating the development and evaluation of dialogue policies.

- Providing a broad collection of compatible datasets and state-of-the-art models.

## 2 Related work

While Rasa (Bocklisch et al., 2017), NeMo (Kuchaiev et al., 2019) and DialogueStudio (Zhang et al., 2023) provide unified data formats, they do not have RL tools or user simulators for interactive training and evaluation of dialogue systems. ParlAI (Miller et al., 2017) includes a *reward* attribute in their unified format, but without accessible RL tools. PyDial (Ultes et al., 2017) and the predecessors of ConvLab-3 (Lee et al., 2019; Zhu et al., 2020b) provide reinforcement learning toolkits, however they lack a unified format and thus the possibility to study generalization across datasets. Moreover, PyDial and previous versions of ConvLab do not provide multiple data-driven user simulators and their training evaluation provides no tools for in-depth analysis. In addition, none of the above toolkits provide a sufficient set of state-of-the-art models for the different components in a TOD system.

## 3 Unified Format

In our unified format, a dataset consists of (1) an **ontology** that defines the annotation schema, (2) **dialogues** with transformed annotations, and (3) a

| Dataset             | Dataset Annotations |      |      |       |     |          |
|---------------------|---------------------|------|------|-------|-----|----------|
|                     | Goal                | DA-U | DA-S | State | API | Database |
| Camrest (2016)      | ✓                   | ✓    | ✓    | ✓     |     | ✓        |
| WOZ 2.0 (2017)      |                     | ✓    |      | ✓     |     |          |
| KVRET (2017)        |                     | ✓    |      | ✓     | ✓   |          |
| DailyDialog (2017)  |                     | ✓    |      |       |     |          |
| Taskmaster-1 (2019) |                     | ✓    | ✓    | ✓     |     |          |
| Taskmaster-2 (2019) |                     | ✓    | ✓    | ✓     |     |          |
| MultiWOZ 2.1 (2020) | ✓                   | ✓    | ✓    | ✓     |     | ✓        |
| SGD (2020)          |                     | ✓    | ✓    | ✓     | ✓   |          |
| MetaLWOZ (2020)     | ✓                   |      |      |       |     |          |
| CrossWOZ (2020a)    | ✓                   | ✓    | ✓    | ✓     | ✓   | ✓        |
| Taskmaster-3 (2021) |                     | ✓    | ✓    | ✓     | ✓   |          |
| EmoWOZ (2022)       | ✓                   | ✓    | ✓    | ✓     |     | ✓        |

Table 1: Annotations of current unified datasets. DA-U/DA-S is dialogue acts annotation of user/system.

**database** that links to external knowledge sources (see Figure 1).

Typically converting the formats of different datasets is not straightforward, hindering format adaptation of existing and new corpora. However, in ConvLab-3 we provide detailed guidelines and scripts that make the process of format adaptation straightforward and error-free. ConvLab-3 offers a large number of datasets in the unified format as shown in Table 1, whilst also simplifying the process of adding new datasets.

Moreover, as shown in Listing 1, we provide utility functions to process the unified datasets, such as delexicalization, splitting data for few-shot learning, and loading data for specific tasks. Based on the unified format, evaluations of common tasks across models and corpora are standardized, which facilitates comparability. More details of already supported datasets and tasks can be found in Appendix A and B, respectively.

### 3.1 Ontology

Following Budzianowski et al. (2018) and Rastogi et al. (2020), an ontology consists of: (1) Domains and their slots in a hierarchical format. Each slot has a Boolean flag indicating whether it is a categorical slot (whose value set is fixed). (2) All possible intents in dialogue acts. (3) Possible dialogue acts appearing in the dialogues. Each act is comprised of intent, domain, slot, and speaker (i.e., system or user). (4) Template dialogue state. We also provide a natural language description, if available, for each domain, slot, and intent to facilitate few-shot learning (Mi et al., 2022) and domain transfer (Lin et al., 2021b).

```

from convlab.util import *

dataset_name = "multiwoz21"
# load dataset: a dict maps data_split to dialogues
dataset = load_dataset(dataset_name)
# load dataset in a predefined order with a custom
# split ratio for reproducible few-shot experiments
dataset = load_dataset(dataset_name, \
                       split2ratio={"train": 0.01})

# load ontology and database similarly
ontology = load_ontology(dataset_name)
database = load_database(dataset_name)
# query the database with domain and state
state = {"hotel": {"area": "east", \
                  "price range": "moderate"}}
res = database.query("hotel", state, topk=3)

# Example functions based on the unified format
# load the user turns in the test set for NLU task
nlu_data = load_nlu_data(dataset, "test", "user")
# dataset-agnostic delexicalization
dataset, delex_vocab = create_delex_data(dataset)

```

Listing 1: Example usage of unified datasets.

### 3.2 Dialogues

We unify the format of dialogue annotations included in many datasets and commonly used by dialogue models while keeping the original format of annotations that only appear in specific datasets. As we integrate more datasets in the future, we will expand the unified format to include more common annotations.

For a dialogue in the unified format, dialogue-level information includes the dataset name, data split (training or test), unique dialogue ID, involved domains, user goal, etc. Following MultiWOZ (Budzianowski et al., 2018), a user goal has informable slot-value pairs, requestable slots, and a natural language instruction summarizing the goal.

Turn-level information includes speaker, utterance, dialogue acts, state, database result, etc (see Appendix H for an example). Each dialogue act is a list of tuples, each tuple consisting of intent, domain, slot, and value. According to the value, we divide dialogue acts into three groups: (1) **categorical** for slots whose value set is predefined in the ontology (e.g., inform the weekday of a flight). (2) **non-categorical** for slots whose values can not be enumerated (e.g., inform the address of a hotel). (3) **binary** for intents without actual values (e.g., request the address of a hotel). The state is initialized by the template state as defined by the ontology and updated during the conversation, containing slot-value pairs of involved domains. A database result is a list of entities retrieved from the database or other knowledge sources. We list common annotations included in the unified data format and the tasks they support in Appendix B.



Other dataset-specific annotations are retained in their original formats.

### 3.3 Database/API Interface

To unify the interaction with different types of databases, we define a `BaseDatabase` class that has an abstract query function to be customized. The query function takes the current domain, dialogue state, and other custom arguments as input and returns a list of top-k candidate entities. By inheriting `BaseDatabase` and overriding the query function, we can easily access different databases/APIs and retrieve the result with a unified format.

### 3.4 Evaluation

To provide a comparable evaluation setup for all TOD tasks supported by the unified format, we provide unified evaluation scripts. These scripts include commonly used metrics such as: turn accuracy (ACC) and dialogue act F1 score for natural language understanding (NLU) (Zhu et al., 2020b), joint goal accuracy (JGA) and slot F1 score for dialogue state tracking (DST) (Li et al., 2021), BLEU and slot error rate (SER) for natural language generation (NLG) (Wen et al., 2015), BLEU and Combined score (Comb.) for End2End dialogue modeling (Mehri et al., 2019), turn accuracy, slot-value F1 score and SER for user simulators (Lin et al., 2021a, 2022).

## 4 Integrated Models

ConvLab-3 provides a wide array of standard and state-of-the-art models covering all modules in a TOD system. This allows straightforward plug-and-play experimentation when developing a specific module, as well as building TOD systems easily on custom datasets. A model is considered integrated once it implements the corresponding module interface and supports processing datasets in the unified format.

Besides existing models in ConvLab-2 (Zhu et al., 2020b), we integrate new transformer-based models supporting the unified data format, including SetSUMBT (van Niekerk et al., 2021) and TripPy (Heck et al., 2020) for dialogue state tracking (DST), DDPT (Geishauser et al., 2022) and LAVA (Lubis et al., 2020) for policy learning, SC-GPT (Peng et al., 2020) for natural language generation (NLG), and SOLOIST (Peng et al., 2021) with T5 as backbone model (Peng et al., 2022) for end-to-end modeling (End2End). We also integrate

multiple powerful data-driven user simulators (US): TUS (Lin et al., 2021a) that outputs user dialogue acts, GenTUS (Lin et al., 2022) that outputs both user dialogue acts and response, and EmoUS (Lin et al., 2023) that additionally outputs emotions.

In addition, we apply text-generation models to solve the tasks of TOD modules (see Appendix C). We provide a range of models built upon T5 (Raffel et al., 2020), covering NLU, DST, NLG, etc. We also provide an interface to instruct large language models (LLMs) such as ChatGPT and LLaMa (Touvron et al., 2023) to serve as different modules such as user simulators, NLU, DST, NLG, etc. See Heck et al. (2023) for an example of how ChatGPT can be instructed to serve as a DST model. All integrated models are shown in Appendix B.

## 5 Reinforcement Learning Toolkit

The difficulty of building a comprehensive TOD toolkit lies in the fact that it needs to support not only supervised but also reinforcement learning. As shown in Figure 1, this includes functionalities to build configurable agents and user simulators consisting of different modules, an evaluator to provide reward signals, and analysis tools to evaluate the training process and RL algorithms. ConvLab-3 supports the straightforward combination of components with an easy-to-use configuration file, including the definition of the interactive environment given by the choice of user policy and its components, see Appendix G for an example.

The dialogue policy module obtains the semantic information of the DST (and NLU) as input and produces a list of atomic actions  $[(domain_1, intent_1, slot_1), \dots]$  as output, e.g.  $[(hotel, inform, phone), (hotel, inform, addr)]$ , which results in a large action space due to the high number of possible atomic actions and their combinations. As the input is on semantic level while the policy network expects vectorized input, ConvLab-3 provides a `Vectoriser` class that acts as communication module between semantic and vector representation. We treat the `Vectoriser` as an additional pipeline module, which allows straightforward investigation of different vectorization strategies in a plug-and-play fashion. Moreover, policy networks can be used off-the-shelf while only the `Vectoriser` needs to be adapted. ConvLab-3 provides a base `Vectoriser` class that can be easily adapted, as well as common vectorization strategies. In addition, we add the possi-

bility for masking certain actions as in Ultes et al. (2017). This allows controllability of the policy output and facilitates learning during RL due to reduction of the large action space. Moreover, in addition to the on-policy RL algorithms REINFORCE (Sutton et al., 1999) and PPO (Schulman et al., 2017), which are already implemented in ConvLab-2, we provide the state-of-the-art continual RL model DDPT together with state-of-the-art algorithms VTRACE (Espeholt et al., 2018) and CLEAR (Rolnick et al., 2019) for off-policy (Sutton and Barto, 2018) and continual RL (Khetarpal et al., 2022), respectively.

## 5.1 Evaluation Tools

Understanding the policy behaviour allows researchers to fine-tune their algorithm or reward model in an informed manner to improve performance. The analysis of policy behaviour can be done by studying 1) the efficiency of actions, i.e. how many atomic actions are taken in a turn, 2) how the selected intents are distributed in a turn, 3) actual dialogue interactions. The average number of atomic actions is an important indicator of information overload, which a user simulator can handle well in contrast to humans. The intent distribution reveals policy preferences and possible exploitations of imperfect user simulators.

ConvLab-3 is the first toolkit to provide these set of measures and evaluation tools together with the common measurements of task success, return and average number of turns. Moreover, actual dialogues can be observed for in-depth evaluation.

## 6 Supervised Learning Experiments

Conducting supervised learning experiments on multiple TOD datasets is convenient with the unified data format. We believe this feature will encourage researchers to build general dialogue models that perform well on various data as well as to investigate knowledge transfer. In these experiments, we demonstrate the ease of evaluating a model’s knowledge transfer abilities using our unified format. Initially, we pre-train all models on the Schema-Guided Dialogue (SGD) (Rastogi et al., 2020) and Taskmaster-1&2&3 (Byrne et al., 2019, 2021) datasets jointly. These models are then fine-tuned on MultiWOZ 2.1 (Eric et al., 2021) in full-data or low-resource settings. To configure these different training setups, one only needs to make a few changes to the unified dataloader parameters,

| DST      | MultiWOZ 2.1        |                     |                     |                     |                     |              |
|----------|---------------------|---------------------|---------------------|---------------------|---------------------|--------------|
|          | 1%                  |                     | 10%                 |                     | 100%                |              |
|          | JGA ↑               | Slot F1 ↑           | JGA ↑               | Slot F1 ↑           | JGA ↑               | Slot F1 ↑    |
| T5-DST   | 14.5<br><b>22.9</b> | 68.5<br><b>74.9</b> | 35.5<br><b>41.2</b> | 84.8<br><b>87.1</b> | 52.6<br>53.1        | 91.9<br>92.0 |
| SetSUMBT | 7.8<br><b>22.7</b>  | 41.8<br><b>77.2</b> | 37.0<br><b>43.8</b> | 84.4<br><b>88.2</b> | 50.3<br>50.7        | 90.8<br>91.2 |
| NLG      | SER ↓               | BLEU ↑              | SER ↓               | BLEU ↑              | SER ↓               | BLEU ↑       |
| T5-NLG   | 19.0<br><b>9.8</b>  | 20.2<br><b>25.8</b> | 6.9<br><b>5.5</b>   | 31.3<br><b>32.9</b> | 3.7<br>3.5          | 35.8<br>35.8 |
| SC-GPT   | 27.3<br><b>9.5</b>  | 14.1<br><b>26.3</b> | 11.2<br><b>6.9</b>  | 28.4<br>28.6        | 4.8<br>5.3          | 33.6<br>32.1 |
| End2End  | Comb.               | BLEU                | Comb.               | BLEU                | Comb.               | BLEU         |
| SOLOIST  | 19.8<br><b>42.2</b> | 0.4<br><b>10.4</b>  | 48.0<br><b>62.0</b> | 10.0<br><b>15.9</b> | 67.0<br><b>71.4</b> | 16.8<br>17.5 |

Table 2: Comparison between models without pre-training (1st row) and with pre-training (2nd row) in both the low-resource and full-data settings.

as depicted in Listing 1. For low-resource fine-tuning, we set the data ratios of both training and validation set to 1% and 10%.

In the low-resource setting, we observe that pre-training is beneficial, as evidenced in Table 2. Specifically for the end-to-end model SOLOIST, pre-training also proves advantageous in the full-data setting. This may be attributed to the increased complexity of the end-to-end modeling task. These findings emphasize that transfer learning can be successfully implemented in ConvLab-3 in a straightforward way. This enables: (1) developers to leverage knowledge from existing datasets for application in smaller, custom settings; (2) newcomers to explore the capabilities of various models; and (3) experienced researchers to evaluate the generalisability of their proposed methods, as well as to compare them to the available state-of-the-art benchmarks. For an example of joint training across multiple datasets and retrieval based data augmentation, see Appendix D and E.

## 7 Reinforcement Learning Experiments

ConvLab-3 supports a convenient way to run RL training and evaluation supported by the unified format and availability of multiple user simulators. To showcase this, we run transfer learning experiments as well as experiments with multiple user simulators.

### 7.1 Transfer Learning

We utilize the DDPT policy model with VTRACE as algorithm and consider four different data set

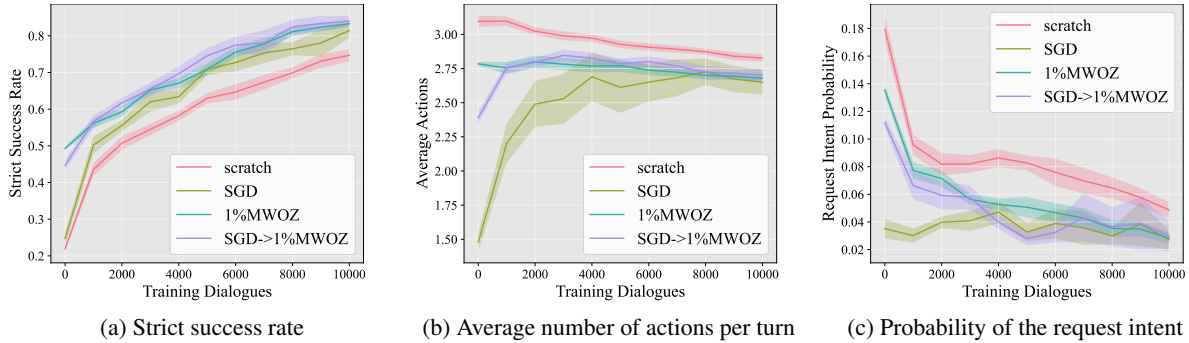


Figure 2: Pre-training then RL training experiments with the DDPT model in interaction with the rule-based simulator. Shaded regions show standard error. Each model is evaluated on 9 different seeds.

| US for training | US for testing |             |             |
|-----------------|----------------|-------------|-------------|
|                 | ABUS           | TUS         | GenTUS      |
| ABUS            | <b>0.93</b>    | 0.71        | 0.56        |
| TUS             | 0.87           | 0.79        | 0.59        |
| GenTUS          | 0.89           | <b>0.86</b> | <b>0.63</b> |

Table 3: The strict success rates of PPO-MLP policies trained on ABUS, TUS, and GenTUS when evaluated with various user simulators.

scenarios for supervised pre-training: (1) **scratch** that does not use pre-training, (2) **SGD** that pre-trains on SGD, (3) **1% MWOZ** that pre-trains on 1% of MultiWOZ data, and (4) **SGD->1%MWOZ** that pre-trains on SGD data and afterwards 1% of MultiWOZ data. The experiments are conducted on the semantic level, leveraging the rule-based dialogue state tracker and the rule-based user simulator (Schatzmann et al., 2007a) of ConvLab-3.

The results, depicted in Figure 2, show a similar trend for all models and metrics. Nevertheless, Figure 2(a) reveals that pre-training on SGD does not yield an advantage for the starting performance, compared to training from scratch, while it leads to better results by the end of training. Moreover, the number of actions taken in a turn and the probability of taking a request intent, shown in Figure 2(b) and (c), is initially much lower for the model trained on SGD only. This indicates that the behaviour learned from SGD differs significantly from the behaviour on MultiWOZ. Refer to Appendix F for more results and experiments. Our unique evaluation tools thus provides essential insights into both metrics and the behaviour of the agent.

## 7.2 Evaluation across Different User Simulators

To enable a policy to generalize to diverse user behaviour, it’s crucial to train and evaluate policy models across various user simulators. ConvLab-3 not only offers state-of-the-art data-driven user simulation models but also a configurable interactive environment for evaluation and reinforcement learning, as illustrated in Figure 1. In these experiments, we utilise a multi-layer perceptron (MLP) policy trained with the PPO algorithm, using three distinct user simulators: ABUS (Schatzmann et al., 2007b), TUS, and GenTUS. We then evaluate the resulting policies using each of these simulators.

The results, listed in Table 3, show that the policy trained with ABUS excels only in ABUS evaluations, while the GenTUS-trained policy outperforms others in GenTUS and TUS evaluations, but performs slightly worse than ABUS-trained policies in ABUS evaluations. This result highlights the importance of cross-US training and evaluation to show the generalizability of the dialogue policy. Conducting such experiments is made straightforward in ConvLab-3, as the user simulator model can be easily changed within the configuration file.

## 8 Conclusion

In this paper, we present the dialogue system toolkit ConvLab-3, which puts a large number of datasets under one umbrella through our proposed unified data format. The usage of the unified format facilitates comparability and significantly reduces the implementation cost required for conducting experiments on multiple datasets. In addition, we provide recent powerful models for all components of a dialogue system and provide a convenient RL toolkit which enables researchers to easily build, train, analyze and evaluate dialogue systems.

We showcase the advantages of the unified format and RL toolkit in a large number of experiments, ranging from pre-training to RL training. The release of ConvLab-3 supports the community in developing the next generation of task-oriented dialogue systems.

## 9 Limitations

As ConvLab-3 is built for text-based TOD systems, we do not currently provide support for speech. One solution for this is the usage of speech recognition and text-to-speech interfaces such as Whisper (Radford et al., 2023) and WaveNet (van den Oord et al., 2016). Secondly, while we provide several datasets in the unified format together with conversion scripts, the conversion of a new dataset still requires manual effort such as normalizing ontologies and transforming dialogue annotations. Lastly, ConvLab-3, currently, only supports the commonly used hierarchical dialogue state representation (Budzianowski et al., 2018) but not yet state representations such as the graph-based state (Andreas et al., 2020) and tree-structure state (Cheng et al., 2020). We consider these limitations as future work to further improve our toolkit.

## Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2021ZD0113304), the National Science Foundation for Distinguished Young Scholars (with No. 62125604), the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096), and sponsored by Tsinghua-Toyota Joint Research Fund.

This work was also supported by the DYMO project which has received funding from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). In addition, it was supported by an Alexander von Humboldt Sofja Kovalevskaja Award endowed by the German Federal Ministry of Education and Research. Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf and the Google Cloud Platform.

## References

- Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitriy Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. [Task-oriented dialogue as dataflow synthesis](#). *Transactions of the Association for Computational Linguistics*, 8:556–571.
- Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. [Rasa: Open source language understanding and dialogue management](#). *arXiv preprint arXiv:1712.05181*.
- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. [MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.
- Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Kale. 2021. [TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 671–680, Online. Association for Computational Linguistics.
- Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. [Taskmaster-1: Toward a realistic and diverse dialog dataset](#). 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 4516–4525, Hong Kong, China. Association for Computational Linguistics.
- Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghda, and Anders Johansen. 2020. [Conversational semantic parsing for dialog state tracking](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

- Mihail Eric, Nicole Chartier, Behnam Hedayatnia, Karthik Gopalakrishnan, Pankaj Rajan, Yang Liu, and Dilek Hakkani-Tur. 2021. [Multi-sentence knowledge selection in open-domain dialogue](#). In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 76–86, Aberdeen, Scotland, UK. Association for Computational Linguistics.
- Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. [MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking base-lines](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.
- Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. [Key-value retrieval networks for task-oriented dialogue](#). In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany. Association for Computational Linguistics.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. [IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR.
- Shutong Feng, Nurul Lubis, Christian Geischauser, Hsien-chin Lin, Michael Heck, Carel van Niekerk, and Milica Gasic. 2022. [EmoWOZ: A large-scale corpus and labelling scheme for emotion recognition in task-oriented dialogue systems](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4096–4113, Marseille, France. European Language Resources Association.
- Christian Geischauser, Carel van Niekerk, Hsien-chin Lin, Nurul Lubis, Michael Heck, Shutong Feng, and Milica Gašić. 2022. [Dynamic dialogue policy for continual reinforcement learning](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 266–284, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geischauser, Hsien-chin Lin, Carel van Niekerk, and Milica Gasic. 2023. [ChatGPT for zero-shot dialogue state tracking: A solution or an opportunity?](#) In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 936–950, Toronto, Canada. Association for Computational Linguistics.
- Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geischauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. [TripPy: A triple copy strategy for value independent neural dialog state tracking](#). In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.
- Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2022. [Towards continual reinforcement learning: A review and perspectives](#). *Journal of Artificial Intelligence Research*, 75:1401–1476.
- 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*.
- 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.
- Jinchao Li, Baolin Peng, Sungjin Lee, Jianfeng Gao, Ryuichi Takanobu, Qi Zhu, Minlie Huang, Hannes Schulz, Adam Atkinson, and Mahmoud Adada. 2020. [Results of the multi-domain task-completion dialog challenge](#). In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, Eighth Dialog System Technology Challenge Workshop*.
- Jinchao Li, Qi Zhu, Lingxiao Luo, Lars Liden, Kaili Huang, Shahin Shayandeh, Runze Liang, Baolin Peng, Zheng Zhang, Swadheen Shukla, Ryuichi Takanobu, Minlie Huang, and Jianfeng Gao. 2021. [Multi-domain task-oriented dialog challenge ii at dstc9](#). In *AAAI-2021 Dialog System Technology Challenge 9 Workshop*.
- Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. [DailyDialog: A manually labelled multi-turn dialogue dataset](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Hsien-Chin Lin, Shutong Feng, Christian Geischauser, Nurul Lubis, Carel van Niekerk, Michael Heck, Benjamin Ruppik, Renato Vukovic, and Milica Gašić. 2023. [Emous: Simulating user emotions in task-oriented dialogues](#). In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23*, page 2526–2531, New York, NY, USA. Association for Computing Machinery.
- Hsien-chin Lin, Christian Geischauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gasic. 2022. [GenTUS: Simulating user behaviour and language in task-oriented dialogues with](#)

- generative transformers. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–282, Edinburgh, UK. Association for Computational Linguistics.
- Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geischauser, Michael Heck, Shutong Feng, and Milica Gasic. 2021a. [Domain-independent user simulation with transformers for task-oriented dialog systems](#). In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.
- Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021b. [Leveraging slot descriptions for zero-shot cross-domain dialogue StateTracking](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648, Online. Association for Computational Linguistics.
- 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.
- Nurul Lubis, Christian Geischauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gasic. 2020. [LAVA: Latent action spaces via variational auto-encoding for dialogue policy optimization](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 465–479, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. [Structured fusion networks for dialog](#). In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177, Stockholm, Sweden. Association for Computational Linguistics.
- Fei Mi, Yitong Li, Yasheng Wang, Xin Jiang, and Qun Liu. 2022. [CINS: comprehensive instruction for few-shot learning in task-oriented dialog systems](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. [ParLAI: 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.
- Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. [Neural belief tracker: Data-driven dialogue state tracking](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.
- Baolin Peng, Michel Galley, Pengcheng He, Chris Brockett, Lars Liden, Elnaz Nouri, Zhou Yu, Bill Dolan, and Jianfeng Gao. 2022. [Godel: Large-scale pre-training for goal-directed dialog](#). *arXiv preprint arXiv:2206.11309*.
- Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayan-deh, Lars Liden, and Jianfeng Gao. 2021. [Soloist: Building task bots at scale with transfer learning and machine teaching](#). *Transactions of the Association for Computational Linguistics*, 9:807–824.
- Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujuan Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. [Few-shot natural language generation for task-oriented dialog](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. [Robust speech recognition via large-scale weak supervision](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.
- 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.
- Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. [Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- 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.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. [Experience replay for continual learning](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

- Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. [Agenda-based user simulation for bootstrapping a POMDP dialogue system](#). In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.
- Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007b. [Agenda-based user simulation for bootstrapping a POMDP dialogue system](#). In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv:1707.06347*.
- Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning: an introduction*, second edition. The MIT Press.
- Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. [Policy gradient methods for reinforcement learning with function approximation](#). In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *arXiv preprint arXiv:2307.09288*.
- 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.
- Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. [Wavenet: A generative model for raw audio](#). In *Arxiv*.
- Carel van Niekerk, Andrey Malinin, Christian Geisshauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gasic. 2021. [Uncertainty measures in neural belief tracking and the effects on dialogue policy performance](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7901–7914, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [MiniLMv2: Multi-head self-attention relation distillation for compressing pre-trained transformers](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. [Conditional generation and snapshot learning in neural dialogue systems](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2153–2162, Austin, Texas. Association for Computational Linguistics.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. [Semantically conditioned LSTM-based natural language generation for spoken dialogue systems](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.
- Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. [The Hidden Information State Approach to Dialog Management](#). In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–149–IV–152.
- Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. [Glm-130b: An open bilingual pre-trained model](#). *arXiv preprint arXiv:2210.02414*.
- Jianguo Zhang, Kun Qian, Zhiwei Liu, Shelby Heinecke, Rui Meng, Ye Liu, Zhou Yu, Huan Wang, Silvio Savarese, and Caiming Xiong. 2023. [Dialogstudio: Towards richest and most diverse unified dataset collection for conversational ai](#).
- Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020a. [CrossWOZ: A large-scale Chinese cross-domain task-oriented dialogue dataset](#). *Transactions of the Association for Computational Linguistics*, 8:281–295.
- Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Akanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020b. [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.

## A Annotations of current unified datasets

The statistics and annotations of already supported datasets are listed in Table 4.

| Dataset                              | Statistics |            |         | Dataset Annotations |      |      |       |            |          |
|--------------------------------------|------------|------------|---------|---------------------|------|------|-------|------------|----------|
|                                      | #Dialogues | Avg. Turns | Domains | Goal                | DA-U | DA-S | State | API Result | Database |
| Camrest (Wen et al., 2016)           | 676        | 10.8       | 1       | ✓                   | ✓    | ✓    | ✓     |            | ✓        |
| WOZ 2.0 (Mrkšić et al., 2017)        | 1200       | 7.4        | 1       |                     | ✓    |      | ✓     |            |          |
| KVRET (Eric et al., 2017)            | 3030       | 5.3        | 3       |                     | ✓    |      | ✓     | ✓          |          |
| DailyDialog (Li et al., 2017)        | 13118      | 7.9        | 10      |                     | ✓    |      |       |            |          |
| Taskmaster-1 (Byrne et al., 2019)    | 13175      | 21.2       | 6       |                     | ✓    | ✓    | ✓     |            |          |
| Taskmaster-2 (Byrne et al., 2019)    | 17303      | 16.9       | 7       |                     | ✓    | ✓    | ✓     |            |          |
| MultiWOZ 2.1 (Eric et al., 2020)     | 10438      | 13.7       | 8       | ✓                   | ✓    | ✓    | ✓     |            | ✓        |
| Schema-Guided (Rastogi et al., 2020) | 22825      | 20.3       | 45      |                     | ✓    | ✓    | ✓     | ✓          |          |
| MetaLWOZ (Li et al., 2020)           | 40203      | 10.4       | 51      | ✓                   |      |      |       |            |          |
| CrossWOZ (Zhu et al., 2020a)         | 6012       | 16.9       | 6       | ✓                   | ✓    | ✓    | ✓     | ✓          | ✓        |
| Taskmaster-3 (Byrne et al., 2021)    | 23757      | 20.1       | 1       |                     | ✓    | ✓    | ✓     | ✓          |          |
| EmoWOZ (Feng et al., 2022)           | 11434      | 14.6       | 8       | ✓                   | ✓    | ✓    | ✓     |            | ✓        |

Table 4: Statistics and annotations of current unified datasets. DA-U/DA-S is dialogue acts annotation of user/system.

## B Tasks and models supported by the unified data format

Tasks and models already supported by the unified data format are shown in Table 5.

## C Example Serialized Dialogue Acts and State

The example of serialized dialogue acts and states is shown in Table 6.

## D Joint Training

In this experiment, we investigate the effect of training a model on multiple datasets jointly instead of separately. For joint training, we merge MultiWOZ, SGD, and Taskmaster datasets into one and train a single model, which requires the model to handle datasets with different ontologies. Intuitively, the advantage of joint training is that knowledge transfer is bi-directional and persists for the whole training period, while the disadvantage is that there may be inconsistent labels for similar inputs on different datasets, potentially confusing the models.

To avoid confusion, for T5-NLU, T5-DST, and T5-NLG, we prepend the dataset name to the original input to distinguish data from different datasets. For SetSUMBT, we only predict the state of the target dataset. Since SGD may have several services for one domain, we normalize the service name to the domain name (e.g., Restaurant\_1 to Restaurant) when evaluating NLU and DST. However, similar slots of different services (e.g., city and location) will still confuse the model. While further normalization may help, we are aiming to compare independent training and joint training instead of achieving SOTA performance. For Taskmaster-1/2/3, we evaluate each sample with

the corresponding ontology and then calculate the metrics on all test samples of three datasets. In addition, on SGD and Taskmaster, we build pseudo user goals for TUS and GenTUS by accumulating constraints and requirements in user dialogue acts during conversations.

We compare independent training and joint training in Table 7. MultiWOZ, SGD, and Taskmaster have 8K, 16K, and 43K dialogues for training respectively. Joint training on these datasets does not lead to substantial performance drops in most cases, indicating that models have sufficient capacity to encode knowledge of different datasets simultaneously. However, joint training does not always improve performance either. It consistently improves the End2End model SOLOIST but makes no difference to T5-NLU. For other models, the gains vary with the dataset. Associating with the previous pre-training-then-fine-tuning experiment, we think the difference may be attributed to the varying task complexity on different datasets. When the original data of a certain dataset are sufficient for a model to solve the task, including other datasets via joint training may not bring further benefit.

## E Retrieval Augmentation

We further explore transferring knowledge from other datasets through retrieval-based data augmentation. Here we only consider the single-turn NLU task where the input is an utterance since utterance-level similarity is easier to model than dialogue-level similarity. For each utterance in the target dataset, we retrieve the top-k ( $k \in \{1, 3\}$ ) most similar utterances from other datasets measured by the MiniLMv2 model (Wang et al., 2021) using Sentence Transformers (Reimers and Gurevych, 2019). We then use retrieved samples in two ways:



| Task           | Input                    | Output           | Models                          |
|----------------|--------------------------|------------------|---------------------------------|
| RG             | Context                  | Response         | T5RG, LLMs                      |
| Goal2Dial      | Goal                     | Dialogue         | T5Goal2Dialogue                 |
| NLU            | Context                  | DA-U             | T5NLU, BERTNLU, MILU, LLMs      |
| DST            | Context                  | State            | T5DST, (Set)SUMBT, TripPy, LLMs |
| Policy         | State, DA-U, Database    | DA-S             | DDPT, PPO, PG                   |
| Word-Policy    | Context, State, Database | Response         | LAVA                            |
| NLG            | DA-S                     | Response         | T5NLG, SC-GPT, LLMs             |
| End2End        | Context, Database        | State, Response  | SOLOIST                         |
| User Simulator | Goal, DA-S               | DA-U, (Response) | TUS, GenTUS, EmoUS, LLMs        |

Table 5: Tasks and models supported by the unified data format. RG is response generation without database support. Goal2Dial is generating a dialogue from a user goal. NLU is natural language understanding. BERTNLU, MILU, PPO, PG are from ConvLab-2 (Zhu et al., 2020b). Currently supported LLMs include LLaMA-2 (Touvron et al., 2023), ChatGLM2 (Zeng et al., 2022), and OpenAI models such as ChatGPT (through API).

---

**User:** I am looking for a *cheap* restaurant.  
**System:** Is there a particular area of town you prefer?  
**User:** In the *centre* of town.

---

**DA-U:** [inform][restaurant]([area][centre])  
**State:** [restaurant]([area][centre],[price range][cheap])  
**DA-S:** [recommend][restaurant]([name][Zizzi Cambridge])

---

**System:** I would recommend *Zizzi Cambridge*.

---

Table 6: Example serialized dialogue acts and state. Dialogue acts are in the form of “[intent] [domain] ([slot] [value],...);...”. State is in the form of “[domain] ([slot] [value],...);...”. Multiple items are separated by a semi-colon.

1. Augment training data. Models are trained on both original training data and retrieved data.
2. Additionally input the retrieved samples as in-context examples, including retrieved utterances and their dialogue acts, as shown in Table 8. Different from Liu et al. (2022), the retrieved samples are from other datasets instead of the target dataset and we will train models on augmented samples.

Since different datasets have different ontologies (i.e., definitions of intent, domain, slot), we prepend the corresponding dataset name to an input utterance as in the joint training experiment. We use the T5-NLU model and try two model sizes T5-Small and T5-Large. We fine-tune the models on MultiWOZ using the same settings as in the pre-training-then-fine-tuning experiment.

## F RL Experiments

### F.1 Transfer Learning

In this section, we provide additional plots for the experiments conducted in Section 7. Figure 3 depicts success rate (which is less strict success rate),

average number of turns as well as the average return. Moreover, we show additional intent probabilities. We can observe in Figure 3(e) that the policy pre-trained on only SGD data uses more offer intents initially, which reflects the behaviour in the data set. The probability of the offer intent then decreases whereas the probability for the recommend intent increases during learning.

### F.2 Training with Uncertainty Features

#### F.2.1 Experimental Setup

The simplified vectorizer module in ConvLab-3 makes it possible to easily include dialogue related features that might benefit dialogue policy learning. One such example is given due to the problem of resolving ambiguities in conversations. Humans naturally identify these ambiguities and resolve the uncertainty resulting from them. For a dialogue system to be robust to ambiguity, it is crucial to identify and resolve these uncertainties (van Niekerk et al., 2021).

ConvLab-3 provides the SetSUMBT dialogue belief tracker which achieves SOTA performance in terms of the accuracy of its uncertainty estimates. Using the vectorizer class to incorporate these features we can train a policy using the uncertainty features obtained from SetSUMBT. To illustrate the effectiveness of uncertainty features during RL, we train a PPO policy using these features. The template-based NLG in ConvLab-3 also allows for the inclusion of noise (van Niekerk et al., 2021) in generated responses which allows for uncertainties to arise during the conversation, simulating a more realistic conversation.

#### F.2.2 Result Analysis

Figure 4 (a) reveals that the policy trained using uncertainty features performs at least as well as the

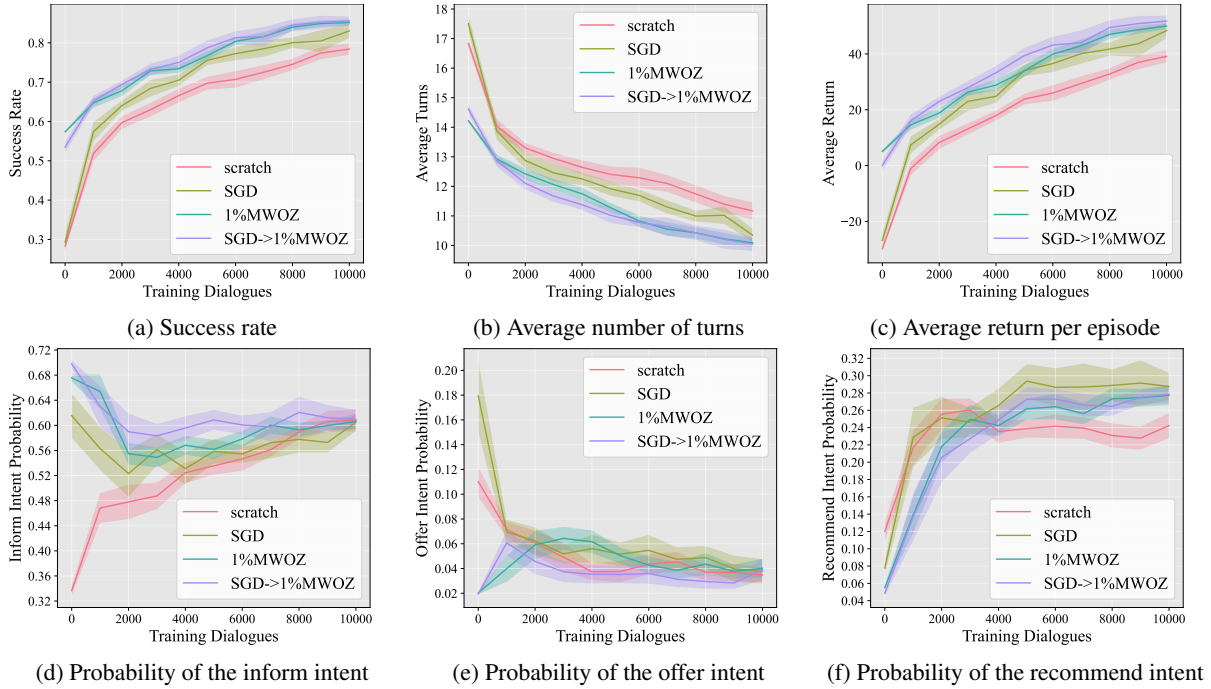


Figure 3: Pre-training then RL training experiments with the DDPT model in interaction with the rule-based simulator. Shaded regions show standard error. Each model is evaluated on 9 different seeds.

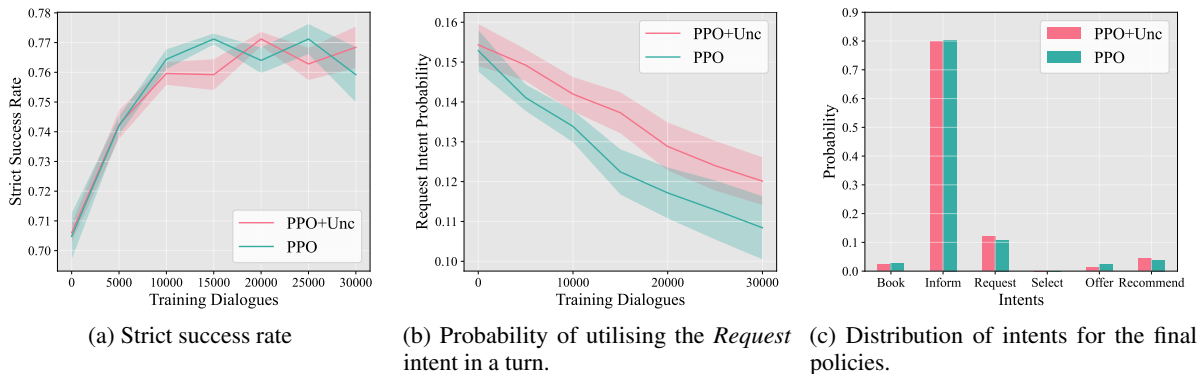


Figure 4: Evaluation of the PPO policy trained combined with a SetSUMBT DST model with and without uncertainty features respectively. The policy is trained in an environment that contains 5% user NLG noise to illustrate the impact of uncertainty.

policy trained without these features. [van Niekerk et al. \(2021\)](#) further showed that the policy trained with uncertainty features performs significantly better in conversation with humans than the policy trained without. This is an indication that the policy using uncertainty features can handle ambiguities in conversation better than the policy without uncertainty modeling. To investigate how this policy resolves uncertainty we analyze the action distributions of the policy using the new RL toolkit evaluation tools, which provide new insights into the behaviour of dialogue policy modules. Figure 4 (b) and (c) show that the policy trained using uncertainty features utilizes significantly more re-

quest actions than the policy without these features. This indicates that the policy aims to resolve uncertainty by requesting information from the user. For instance, if the policy recognizes uncertainty regarding the price range a user has requested, it can resolve this through the use of a request. See [van Niekerk et al. \(2021\)](#) for example dialogues with humans where this can be observed.

## G Configuring the Dialogue Pipeline and User Environment

The modules in the dialogue system pipeline and the user environment for interaction are specified

| NLU      | MultiWOZ 2.1 |         | SGD     |         | Taskmaster |         |
|----------|--------------|---------|---------|---------|------------|---------|
|          | ACC          | F1      | ACC     | F1      | ACC        | F1      |
| T5-NLU   | 77.8         | 86.5    | 45.0    | 58.6    | 81.8       | 73.0    |
|          | 77.5         | 86.4    | 45.2    | 58.6    | 81.8       | 73.0    |
| DST      | JGA          | Slot F1 | JGA     | Slot F1 | JGA        | Slot F1 |
| T5-DST   | 52.6         | 91.9    | 20.1    | 58.5    | 48.5       | 81.1    |
|          | 53.1         | 91.9    | 20.6    | 60.0    | 48.6       | 81.0    |
| SetSUMBT | 50.3         | 90.8    | 20.0    | 58.8    | 24.9       | 65.5    |
|          | 50.8         | 91.0    | 21.1    | 59.2    | 25.3       | 67.0    |
| NLG      | SER ↓        | BLEU    | SER ↓   | BLEU    | SER ↓      | BLEU    |
| T5-NLG   | 3.7          | 35.8    | 11.9    | 29.6    | 2.1        | 51.5    |
|          | 3.2          | 35.6    | 8.3     | 29.9    | 2.0        | 51.3    |
| SC-GPT   | 4.8          | 33.6    | 9.6     | 28.2    | 2.2        | 47.9    |
|          | 3.3          | 33.5    | 6.8     | 29.8    | 1.6        | 47.3    |
| End2End  | Comb.        | BLEU    | Slot F1 | BLEU    | Slot F1    | BLEU    |
| SOLOIST  | 67.0         | 16.8    | 56.9    | 11.2    | 8.5        | 28.0    |
|          | 71.4         | 17.1    | 69.7    | 23.1    | 9.2        | 29.2    |
| US-DA    | ACC          | F1      | ACC     | F1      | ACC        | F1      |
| TUS      | 15.0         | 53.2    | 10.2    | 11.6    | 23.0       | 23.0    |
|          | 32.0         | 62.3    | 13.8    | 15.6    | 22.5       | 22.0    |
| US-NL    | SER ↓        | F1      | SER ↓   | F1      | SER ↓      | F1      |
| GenTUS   | 4.2          | 62.5    | 8.4     | 48.8    | 4.2        | 43.8    |
|          | 3.3          | 49.5    | 8.0     | 48.7    | 3.5        | 43.9    |

Table 7: Comparison of independent training and joint training (1st row vs. 2nd row of each model) on 3 datasets. We normalize the service name to the domain name when evaluating NLU and DST on SGD.

using a configuration file as can be seen in Listing 2.

## H Ontology, Data and Database in the Unified Format

As explained in Section 3, a dataset in the unified format consists of an ontology, dialogues and a database. We depict an example for ontology, dialogues and database results for the MultiWOZ 2.1 dataset in the unified format in Listing 3, 4 and 5, respectively.

**Original input:** user: Yes please, for 8 people at 18:30 on thursday.

**Augmented input:** **tm3** user: Yes, please, four for 8:10pm. => [inform] [movie]([num.tickets][four],[time.showing][8:10 pm]) **tm1** user: Yes, 8PM, please. => [inform][restaurant \_reservation]([time.reservation][8PM]) **sgd** user: Yes please, for 3 people on March 8th at 12:30 pm. => [affirm\_intent] [Restaurants\_1]([[]]);[inform][Restaurants\_1] ([party\_size][3] [date][March 8th],[time][12:30 pm]) **multiwoz21** user: Yes please, for 8 people at 18:30 on thursday.

**Output:** [inform][restaurant]([book day][thursday],[book time][18:30],[book people][8])

Table 8: An example of input augmented by retrieved top-3 samples from other TOD datasets for in-context learning. Dataset names are highlighted.

| T5-NLU            | MultiWOZ 2.1 |      |      |      |      |      |
|-------------------|--------------|------|------|------|------|------|
|                   | 1%           |      | 10%  |      | 100% |      |
| T5-small          | ACC          | F1   | ACC  | F1   | ACC  | F1   |
| Baseline          | 48.1         | 64.6 | 68.8 | 80.6 | 77.8 | 86.5 |
| Pre-trained       | 55.5         | 70.1 | 69.8 | 81.0 | 77.9 | 86.5 |
| <i>Data Aug.</i>  |              |      |      |      |      |      |
| - top1            | 51.4         | 66.8 | 69.0 | 80.6 | 77.5 | 86.5 |
| - top3            | 51.3         | 66.4 | 68.8 | 80.6 | 77.0 | 86.2 |
| <i>In-context</i> |              |      |      |      |      |      |
| - top1            | 44.5         | 61.3 | 68.5 | 80.1 | 77.7 | 86.4 |
| - top3            | 43.6         | 60.8 | 68.1 | 79.8 | 77.5 | 86.4 |
| T5-large          | ACC          | F1   | ACC  | F1   | ACC  | F1   |
| Baseline          | 51.2         | 67.9 | 67.8 | 80.0 | 76.8 | 86.4 |
| Pre-trained       | 56.7         | 71.7 | 69.5 | 81.0 | 76.8 | 86.1 |
| <i>Data Aug.</i>  |              |      |      |      |      |      |
| - top1            | 49.7         | 66.8 | 68.7 | 80.6 | 76.9 | 86.1 |
| - top3            | 48.5         | 66.2 | 68.5 | 80.5 | 76.3 | 85.8 |
| <i>In-context</i> |              |      |      |      |      |      |
| - top1            | 43.4         | 61.3 | 69.1 | 81.0 | 76.5 | 85.9 |
| - top3            | 43.9         | 63.0 | 68.7 | 80.8 | 76.9 | 86.2 |

Table 9: Comparison of different ways to use other TOD datasets: (1) pre-training, (2) retrieving similar samples for data augmentation or (3) in-context learning.

```

{
  "model": {
    "load_path": "from_pretrained",
    "pretrained_load_path": "",
    "use_pretrained_initialisation": false,
    "batchsz": 200,
    "seed": 0,
    "epoch": 100,
    "eval_frequency": 5,
    "process_num": 1,
    "num_eval_dialogues": 20,
    "sys_semantic_to_usr": false
  },
  "vectorizer_sys": {
    "uncertainty_vector_mul": {
      "class_path": "convlab.policy.vector.vector_binary.VectorBinary",
      "ini_params": {
        "use_masking": true,
        "manually_add_entity_names": true,
        "seed": 0
      }
    }
  },
  "nlu_sys": {
    "BertNLU": {
      "class_path": "convlab.nlu.jointBERT.unified_datasets.BERTNLU",
      "ini_params": {
        "mode": "all",
        "config_file": "multiwoz21_all.json",
        "model_file": "https://huggingface.co/ConvLab/bert-base-nlu/resolve/main/bertnlu_unified_multiwoz21_all_context0.zip"
      }
    }
  },
  "dst_sys": {
    "RuleDST": {
      "class_path": "convlab.dst.rule.multiwoz.dst.RuleDST",
      "ini_params": {}
    }
  },
  "sys_nlg": {},
  "nlu_usr": {},
  "dst_usr": {},
  "policy_usr": {
    "GenTUS": {
      "class_path": "convlab.policy.genTUS.stepGenTUS.UserPolicy",
      "ini_params": {
        "model_checkpoint": "convlab/policy/genTUS/unify/experiments/multiwoz21-exp",
        "mode": "language",
        "only_action": false
      }
    }
  },
  "usr_nlg": {}
}

```

Listing 2: Example configuration file for a PPO dialogue policy, binary vectoriser, BertNLU and rule-based DST. The user simulator is GenTUS. We can straightforwardly build different configurations by substituting different modules.

```

{
  "domains": {
    "attraction": {
      "description": "find an attraction",
      "slots": {
        "area": {
          "description": "area to search for attractions",
          "is_categorical": true,
          "possible_values": ["centre", "east", "north", "south", "west"]
        },
        "name": {
          "description": "name of the attraction",
          "is_categorical": false,
          "possible_values": []
        }
      },
      ...
    },
    ...
  },
  "intents": {
    "inform": {"description": "inform the value of a slot"},
    "request": {"description": "ask for the value of a slot"},
    ...
  },
  "state": {
    "attraction": {
      "type": "",
      "name": "",
      "area": ""
    },
    "hotel": {
      "name": "",
      "area": "",
      ...
    },
    ...
  },
  "dialogue_acts": {
    "categorical": [
      "{ 'user': False, 'system': True, 'intent': 'nobook', 'domain': 'hotel', 'slot': 'book day' }",
      "{ 'user': False, 'system': True, 'intent': 'nobook', 'domain': 'restaurant', 'slot': 'book day' }",
      ...
    ],
    "non-categorical": [
      "{ 'user': False, 'system': True, 'intent': 'inform', 'domain': 'attraction', 'slot': 'address' }",
      "{ 'user': False, 'system': True, 'intent': 'inform', 'domain': 'attraction', 'slot': 'choice' }",
      ...
    ],
    "binary": [
      "{ 'user': False, 'system': True, 'intent': 'book', 'domain': 'attraction', 'slot': '' }",
      "{ 'user': False, 'system': True, 'intent': 'book', 'domain': 'hospital', 'slot': '' }",
      ...
    ]
  }
}

```

Listing 3: Example of the unified format ontology for the MultiWOZ 2.1 dataset.

```

[
  {
    "dataset": "multiwoz21",
    "data_split": "train",
    "dialogue_id": "multiwoz21-train-0",
    "original_id": "SNG01856.json",
    "domains": ["hotel", "general"],
    "goal": {
      "description": "You are looking for a place to stay. The hotel should be in the cheap price
range and should be in the type of hotel...",
      "inform": {
        "hotel": {
          "type": "hotel",
          "parking": "yes",
          "price range": "cheap",
          "internet": "yes",
          "book stay": "3|2",
          "book day": "tuesday",
          "book people": "6"
        }
      },
      "request": {
        "hotel": {}
      }
    },
    "turns": [
      {
        "speaker": "user",
        "utterance": "am looking for a place to to stay that has cheap price range it should be in a
type of hotel",
        "utt_idx": 0,
        "dialogue_acts": {
          "categorical": [
            {
              "intent": "inform",
              "domain": "hotel",
              "slot": "price range",
              "value": "cheap"
            }
          ],
          "non-categorical": [
            {
              "intent": "inform",
              "domain": "hotel",
              "slot": "type",
              "value": "hotel",
              "start": 87,
              "end": 92
            }
          ],
          "binary": []
        },
        "state": {
          "attraction": {
            "type": "",
            "name": "",
            "area": ""
          },
          ...
        }
      },
      {
        "speaker": "system",
        "utterance": "Okay, do you have a specific area you want to stay in?",
        "utt_idx": 1,
        "dialogue_acts": {
          "categorical": [],
          "non-categorical": [],
          "binary": [
            {
              "intent": "request",
              "domain": "hotel",
              "slot": "area"
            }
          ]
        },
        "booked": {
          "taxi": [],
          "restaurant": [],
          ...
        }
      },
      ...
    ]
  },
  ...
]

```

Listing 4: Example of the unified format data within the MultiWOZ 2.1 dataset.

```
[
  {
    "address": "124 tenison road",
    "area": "east",
    "internet": "yes",
    "parking": "no",
    "id": "0",
    "name": "a and b guest house",
    "phone": "01223315702",
    "postcode": "cb12dp",
    "price": {
      "double": "70",
      "family": "90",
      "single": "50"
    },
    "pricerange": "moderate",
    "stars": "4",
    "takesbookings": "yes",
    "type": "guesthouse",
    "Ref": "00000000"
  },
  ...
]
```

Listing 5: Example of database query result when searching for a moderately priced hotel in the east from the MultiWOZ unified format database.

# FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge

Farima Fatahi Bayat<sup>1\*</sup>, Kun Qian<sup>2</sup>, Benjamin Han<sup>2</sup>, Yisi Sang<sup>2</sup>, Anton Belyi<sup>2</sup>,  
Samira Khorshidi<sup>2</sup>, Fei Wu<sup>2</sup>, Ihab F. Ilyas<sup>2</sup>, Yunyao Li<sup>2</sup>

University of Michigan - Ann Arbor<sup>1</sup>, Apple<sup>2</sup>

farimaf@umich.edu, {kunqian, ben.b.han, yisi\_sang, a\_belyy}@apple.com

{samiraa, fwu7, iilyas, yunyaoli}@apple.com

## Abstract

Detecting factual errors in textual information, whether generated by large language models (LLM) or curated by humans, is crucial for making informed decisions. LLMs’ inability to attribute their claims to external knowledge and their tendency to hallucinate makes it difficult to rely on their responses. Humans, too, are prone to factual errors in their writing. Since manual detection and correction of factual errors is labor-intensive, developing an automatic approach can greatly reduce human effort. We present FLEEK, a prototype tool that automatically extracts factual claims from text, gathers evidence from external knowledge sources, evaluates the factuality of each claim, and suggests revisions for identified errors using the collected evidence. Initial empirical evaluation on fact error detection (77-85% F1) shows the potential of FLEEK. A video demo of FLEEK can be found at <https://youtu.be/NapJFUlkPdQ>.

## 1 Introduction

While textual information offers a convenient and efficient means of communication, it is critical to acknowledge its potential for misuse or unintended consequences. False or misleading information spreads easily over online platforms (Webb et al., 2016). Additionally, the emergence of powerful Large Language Models (LLMs) such as GPT models<sup>1</sup>, Vicuna (Chiang et al., 2023), and Alpaca (Taori et al., 2023) have introduced a new avenue for knowledge-seeking inquiries. These models, however, have a tendency to hallucinate and provide creative and fluent responses that are not factually accurate (Pan et al., 2023). The limitation of LLMs to attribute their responses to external valid evidence makes it challenging to trust their responses. Therefore, having a robust fact-checking

mechanism is of paramount importance to ensure the integrity and accuracy of information.

Previous works (Zhong et al., 2019; Liello et al., 2022; Liu et al., 2020) and systems like FACTGPT<sup>2</sup>, typically formulates the fact verification as a classification task where the input consists of the evidence sentence(s) and the claim, and the output is a label indicating the veracity of the entire claim as SUPPORTED, REFUTED, or IRRELEVANT. As a concrete example, if the claim is “*United States is in North America and has 51 states*”, then a sentence-level classification task would classify this claim as incorrect since there are 50 states in the United States. However, this claim actually contains one valid sub-claim: “*United States is in North America*” ✓, and one false sub-claim: “*United States has 51 States*” ✗. Providing a single label stating that this claim is not supported or a single score indicating its factual accuracy is not helpful. It would still require users to manually identify text spans corresponding to potential incorrect facts, generate search queries to gather evidence from the open web, and ultimately make a decision based on multiple pieces of evidence.

In this work, we present FLEEK (Factual Error detection and correction with Evidence Retrieved from external Knowledge), an intelligent and model-agnostic tool designed to support end users (e.g. human graders) in fact verification and correction. Our tool features an intuitive and user-friendly interface, capable of automatically identifying potential verifiable facts from input text. It generated questions for each fact and queries both curated knowledge graphs and the open web to collect evidence. Our tool then verifies the correctness of the facts using the gathered evidence, and suggests revisions to the original text.

Our verification process is naturally interpretable since the extracted facts, generated questions, and retrieved evidence all directly reflect which infor-

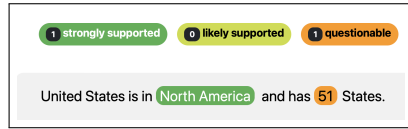
\*Work done while the author was an intern at Apple.

<sup>1</sup><https://platform.openai.com/docs/models>

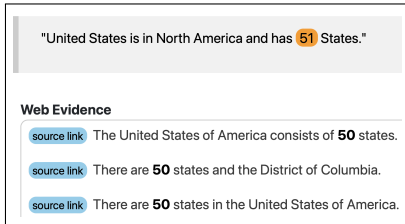
<sup>2</sup><https://factgpt-fe.vercel.app/>



mation units contribute to the verification process. For the example mentioned above, FLEEK would highlight verifiable facts with different colors indicating their factuality levels (see Figure 1(a)), and these clickable highlights can open a dialog that further lists evidence retrieved to support or refute each claim (see Figure 1(b)).



(a) Factuality annotations by FLEEK



(b) A clickable questionable fact

Figure 1: Screenshots of FLEEK

To the best of our knowledge, FLEEK is the first verification and correction system that provides fact-level decisions, attributes them with evidence from online sources of information, and proposes factual revisions.

## 2 Methodology

Figure 2 shows the overall architecture of FLEEK. Basically, FLEEK can perform two tasks: *Fact verification* and *Fact Revision*. Next, we describe the methodology used to enable the two tasks.

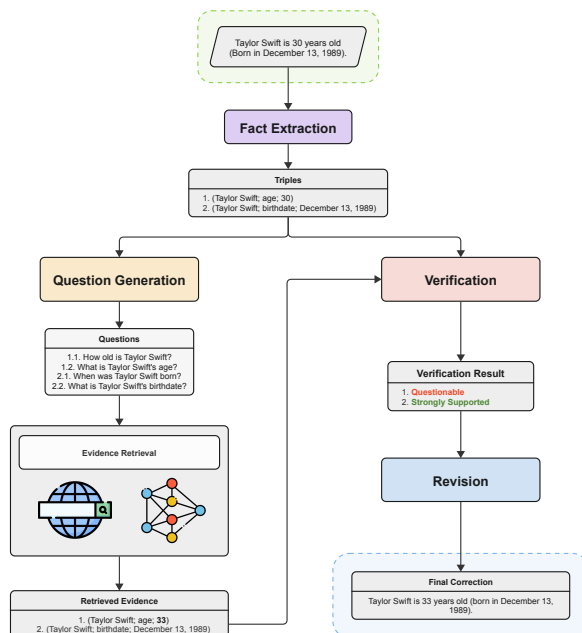


Figure 2: FLEEK verification and revision framework.

## 2.1 Fact Verification

Given an input passage  $p$ , we split it into a set of sentences  $\{s_1, \dots, s_i\}$ . We then verify each sentence using the sequential pipeline described below.

### 2.1.1 Fact Extraction

In this work, we define a *fact* as a unit of information that (1) describes a certain entity or (2) captures the relation between two entities (3) describes an event. Each fact consists of a subject, a predicate, and at least one object. We use the semi-structured triple format to represent such a fact. Our goal is to break a sentence into a set of triples such that each triple represents a verifiable piece of information. This way, we can provide more fine-grained verification details for each sentence. To exhaustively extract facts, we consider two triple formats:

**Flat Triple:** For binary predicates, i.e., predicates with one object, we represent the fact in the form of  $(Subject; Predicate; Object)$ . For example, the triple representation of the fact “Taylor Swift is 30 years old.” is  $(Taylor\ Swift; age; 30\ years\ old)$ .

**Extended Triple Ilyas et al. (2022):** For  $n$ -ary predicates where  $n > 2$ , i.e., predicates with multiple objects, we utilize the *extended triple* format to capture the relations between fact constituents. The extended triple format is  $(Subject; Predicate; Predicate\_ID; Predicate\_attribute; Object)$  where  $Predicate\_ID$  is an artificial predicate identifier,  $Predicate\_attribute$  is the name of the predicate’s attribute, and  $Object$  is the attribute’s value. For instance, the representation of the sentence “Taylor Swift moved to Nashville at the age of 14.” is:  $(Taylor\ Swift; moved; move\_ID; place; Nashville)$   $(Taylor\ Swift; moved; move\_ID; age; 14)$ .

For an input sentence  $s_i$ , the task of Fact Extraction is to extract flat triples  $T_f = \{t_{f_1}, \dots, t_{f_m}\}$  and extended triples  $T_e = \{t_{e_1}, \dots, t_{e_n}\}$ . These triples are extracted from the sentence using an open information extraction format, with each triple representing a single predicate attribute. The final output of this component is:  $T = T_f \cup T_e$ .

To extract these triples, we came up with five challenging human demonstrations such that, for an input sentence, they include different combinations of flat and extended triples. We prompt two instructable LLMs to obtain such triples. More details on LLMs utilized for this task, along with an in-depth analysis of the errors they generate, is provided in section 4.

## 2.1.2 Question Generation (QGen)

Given the output of the Fact Extraction component  $T$ , the task of Question Generation is to generate questions for each  $t \in T$  such that the answer to the question is the *Object* part of  $t$ . In this way, various answers retrieved from different sources can be used to verify each triple  $t$ . Depending on the format of triple  $t$  (flat or extended), we introduce two different question generation paradigms.

**Type-aware Question Generation (TQGen).** Consider the triple (*Taylor Swift; birthdate; 1989*). If the output of the QGen component is: “*When was Taylor Swift born?*”, the retrieved evidence would probably be the exact birthdate. To generate a question as specific and close to the answer as desired, we propose a type-aware question generation approach. Using TQGen, the generated question will be: “*In which year was Taylor Swift born?*”. To this end, we adopt the *Chain-of-Thought* paradigm. This involves two steps: we instruct our model to first find the “*type*” of the *Object* in the input triple, and then generate a question conditioned on the obtained type information. TQGen guides the subsequent information retrieval component to target and retrieve the exact fact that we aim to verify. Prompting LLMs with two human demonstrations was sufficient for this task.

**Context-driven Question Generation (CQGen).** In addition to generating a precise type-aware question, we need to provide context for extended triples so that the retrieved evidence corresponds to the exact situation that requires verification. Consider the extended triples mentioned earlier, (*Taylor Swift; moved; move\_ID; place; Nashville*) (*Taylor Swift; moved; move\_ID; age; 14*) where the focus is on generating a question for the first triple. If we only feed the first triple to QGen, the output would not consider the time when the relocation happened. To generate a context-driven question, we need to also feed the second triple, the *context triple*, to Context-driven QGen (CQGen). The output of CQGen in this example is “*To which city did Taylor Swift move to at the age of 14?*”. For this task, we prompt LLMs with two examples.

## 2.1.3 Evidence Retrieval

The generated questions will be sent to two retrieval systems: a knowledge graph (KG)-based system and a web-based system.

**Knowledge Graph-based:** We send the question generated for each triple  $t$  to our KG question answering (KGQA) system and collect the retrieved

short answers. The answer can either be a single value (e.g., birth date, birthplace) or a list (e.g., profession, spouses). The ensuing entailment decision is derived differently for these two forms of answers (more details in Section 2.1.4).

**Web-based:** Similarly, we also submit the same question(s) to our web search engine (Web Search). We then take the top-k (e.g., 5) web passages returned for each question and combine them to create a consolidated set of answers. Additionally, Web Search is able to highlight the short answer  $a$  for each retrieved passage  $p$ . The final retrieval list from Web Search is in the format  $[(p_1, a_1), (p_2, a_2), \dots, (p_k, a_k)]$ .

## 2.1.4 Verification

Given the triple representation of a fact  $t$ , the set of KG answers  $A_{kg} = \{a_1, a_2, \dots\}$ , and the set of Web answers  $A_w = \{(p_1, a_1), (p_2, a_2), \dots\}$ , the task is to decide whether  $t$  is supported by the set of retrieved evidence. This involves two steps:

**Step 1 - Verify against KG answers.** Based on our observation, when the evidence retrieved from the KG is a singular value, the expected answer to the question is most likely to also be a single value (e.g. city of birth). Therefore when  $|A_{kg}| = 1$ , we classify the fact as “*Strongly Supported*” if it is entailed by the answer, and “*questionable*” otherwise. However, if the KG answer is a list, we classify each answer in  $A_{kg}$  as either “*supporting*” or “*not supporting*” based on whether it entails the fact. In this case, due to the limited coverage of facts in KG (Dong et al., 2014; Peng et al., 2023), we verify the fact  $t$  against web answers as well.

**Step 2 - Verify against Web answers.** In case the KG answer is empty or a list, web answers will be also used to make a decision. We classify the answers in  $A_w$  as either “*supporting*” or “*not supporting*” evidence. Finally, the fact is labeled as “*Likely Supported*” if our system finds at least one “*supporting*” evidence and “*Questionable*” otherwise. In what follows, we describe how perform evidence classification.

**Triple Entailment.** For every triple  $t$ , we have a set of retrieved answers  $A = A_{kg} \cup \{a_i | a_i \in A_w\}$ . Our task is to classify each answer as either “*supporting*” or “*not supporting*”. To this end, we construct an evidence triple  $t_e$  by replacing the object part of the triple with the short answer retrieved. Therefore, for each  $a \in A$  and triple  $t = (S; P; O)$ , the corresponding evidence triple is  $t_e = (S; P; a)$ . If the claim triple  $t = (S; P; Pid; P_attr; O)$

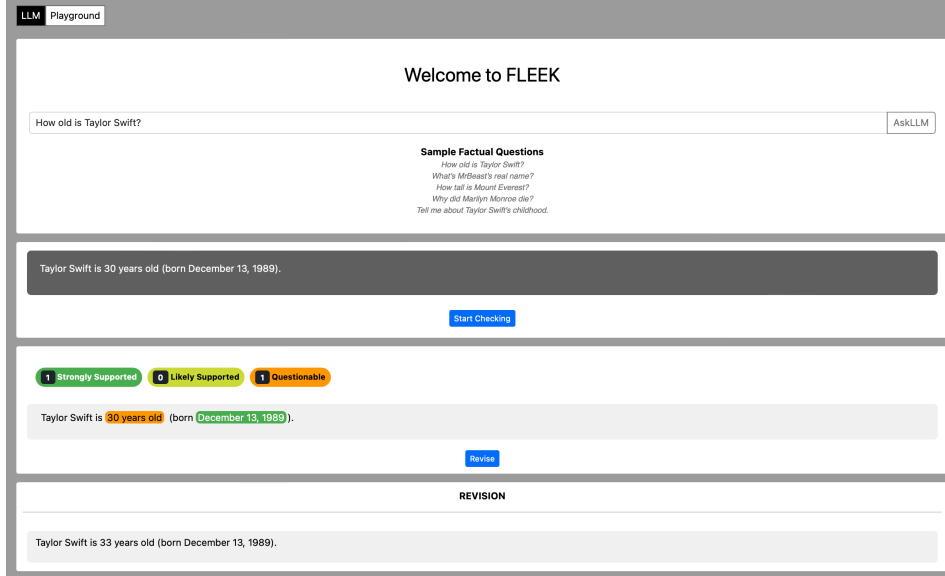


Figure 3: FLEEK LLM view.

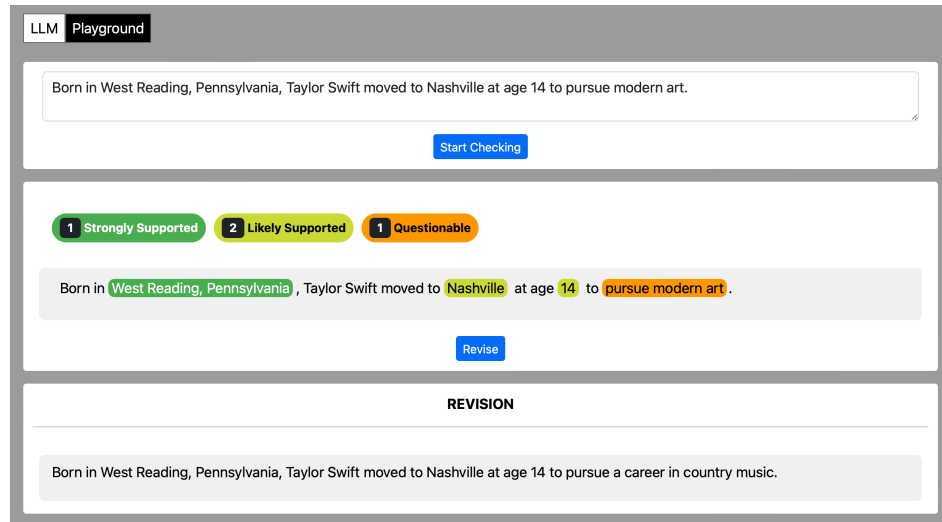


Figure 4: Playground view.

is extended, the corresponding evidence triple is  $t_e = (S; P; Pid; P\_attr; a)$ . The claim and its corresponding evidence triple are then used to form a prompt and fed to LLM to make a final decision.

## 2.2 Fact Revision

The Fact Revision module aims to correct a questionable fact triple stated in an input sentence into its corrected version while preserving everything else stated in the sentence. More specifically, let  $s$  be a sentence containing a questionable triple  $t_{src}$ , i.e.,  $s \models t_{src}$  (i.e.,  $s$  entails fact  $t_{src}$ ). Let the evidence triple formed by the verification process outlined above be  $t_{dest}$ . The Fact Revision model will thus rewrite  $s$  into  $s'$  such that  $s' \not\models t_{src} \wedge s' \models t_{dest}$ , and  $s' \models t_i$  where  $t_i \neq t_{src}$  is any triple entailed by  $s$ . Following is an example

(the objects of the triples are in bold):

$s = \text{“Taylor Swift is } \mathbf{30} \text{ years old.”}$

$t_{src} = (\text{Taylor Swift}; \text{age}; \mathbf{30})$

$t_{dest} = (\text{Taylor Swift}; \text{age}; \mathbf{33})$

$s' = \text{“Taylor Swift is } \mathbf{33} \text{ years old.”}$

In our implementation, we prompt LLMs with one demonstration to obtain satisfactory results.

## 3 The User Interface of FLEEK

The frontend of FLEEK is built using Angular<sup>3</sup> and Bootstrap UI<sup>4</sup>, which allows for creating dynamic, interactive, and visually appealing user interface. The backend of FLEEK is handled by Django<sup>5</sup>, a Python-based server-side framework

<sup>3</sup><https://angular.io/>

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

<sup>5</sup><https://www.djangoproject.com/>

that facilitates the integration with ML-based libraries. The entry point to the system is the two views, LLM and Playground, shown at Figure 3, which we describe next.

**LLM View.** In this view, the user can check the factual consistency of an LLM (e.g. GPT-3.5) that the user provided (as an endpoint). To interact with FLEEK, first in the **Input Panel** (Figure 3, upper panel), the user can type their query in the question bar, e.g., ‘How old is Taylor Swift?’ (or click one of the sample queries) and hit the AskLLM button. FLEEK would send the query to the LLM (GPT-3 in this example) and render its response in the **Response Panel** (Figure 3, the upper second panel in dark grey). The verification process will kick in once the user hit the Start Checking button. FLEEK verifies the claim(s) made by the LLM by going through the process described in Section 2.1. The verification results are shown in the **Verification Panel** (Figure 3, middle panel). With our design, FLEEK is able to highlight the sub-claims in the original text with different color codes to indicate their factual accuracy categories based on the collected evidence. Additionally, the highlighted spans are clickable, which leads to a detailed dialog containing the evidence associated with the claims (illustrated in Figure 1 and Figure ??). Evidence retrieved from web are accompanied with a source link as well. At the bottom, the user can request FLEEK (hit the Revise button) to revise the original claims using the collected evidence. Based on the evidence retrieved from the KG and web, we can have multiple revision alternatives. Verification results for the example shown in Figure 3 and 4 allow for only one possible revision.

**Playground View.** This view allows the user to verify any specific piece of text of their choosing. This feature empowers users to automatically fact-check tweets, trending news, arbitrary LLM outputs, or even their own writing with just a few clicks. Figure 4 illustrates the view. The user can input their desired text into the designated input panel (scratchpad) and hit the "Start Checking" Button (Figure 4, upper panel). The verification and revision process is the same as in the *LLM View*.

## 4 Evaluation

Previous benchmarks on fact verification (Thorne et al., 2018; Aly et al., 2021) provide a single decision for the entire claim based on the retrieved evidence. However, in this work, we introduce fine-

grained fact verification with attribution to external knowledge. As this is the first study on this task, there exist no benchmarks for evaluating FLEEK’s performance. Next, we conduct preliminary experiments using manually created evaluation data.

### 4.1 Evaluation Data Creation

Our system has two use cases. The first one is to verify the responses generated by LLMs (in this case, GPT-3). To evaluate our system’s performance, we selected 50 questions from WikiQA (Wikipedia open-domain Question Answering) test set (Yang et al., 2015) and collected their corresponding GPT-3 responses. We then manually annotated each response using the following steps: (1) identify the facts within the response, (2) label each fact as “*Strongly Supported*”, “*Likely Supported*”, or “*Questionable*”, (3) accompany each fact with an evidence set, particularly the questionable facts. We call this dataset *Bench<sub>LLM</sub>*. Each instance in the *Bench<sub>LLM</sub>* contains the annotated GPT-3-generated response.

The second use case is to verify an arbitrary input text. To create evaluation data that suits this task, we target the introduction section of Wikipedia pages. To partially perturb sentences and create incorrect facts, we sample 50 random sentences with at least one hyperlink. Then, we retrieve the hyperlink’s corresponding entity from Wikidata<sup>6</sup>, find the entity’s type (*instance of* property), and retrieve candidate entities with the same type. Finally, we perturb the sentence by replacing the original hyperlink with one randomly selected entity within the candidate list. After perturbation, we annotate the sentence the same way that we created *Bench<sub>LLM</sub>*. We call this dataset *Bench<sub>Text</sub>*.

### 4.2 Large Language Models

All FLEEK’s components that facilitate fact verification and correction use few-shot prompting with a large language model. Any model that can learn from in-context demonstrations can be used to instantiate FLEEK. We choose one open-source model, Vicuna (33 billion parameters), and one closed source model, GPT-3 (175 billion parameters), to create two instances of our tool. We call the instance with Vicuna as its large language model FLEEK<sub>Vicuna</sub> and the instance that utilizes GPT-3 as its large language model FLEEK<sub>GPT-3</sub>. We evaluate both instances in the following section.

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

| Instance         | Category           | $Bench_{LLM}$ |       |              | $Bench_{Text}$ |       |              |
|------------------|--------------------|---------------|-------|--------------|----------------|-------|--------------|
|                  |                    | P             | R     | F1           | P              | R     | F1           |
| $FLEEK_{Vicuna}$ | Strongly Supported | 91.66         | 84.61 | 87.99        | 84.61          | 91.66 | 87.99        |
|                  | Likely Supported   | 94.73         | 58.69 | 72.47        | 87.5           | 37.33 | 52.33        |
|                  | Questionable       | 54.54         | 60    | 57.13        | 93.61          | 74.57 | 83.01        |
|                  | Total              | 88.75         | 61.73 | <b>72.81</b> | 90.21          | 56.84 | <b>69.73</b> |
| $FLEEK_{GPT-3}$  | Strongly Supported | 100           | 95.23 | 97.55        | 100            | 100   | 100          |
|                  | Likely Supported   | 93.22         | 61.79 | 74.31        | 95.77          | 79.06 | 86.61        |
|                  | Questionable       | 66.66         | 100   | 79.99        | 87.75          | 69.35 | 77.47        |
|                  | Total              | 89.0          | 76.06 | <b>82.02</b> | 93.28          | 77.16 | <b>84.45</b> |

Table 1: Evaluating two instances of FLEEK on  $Bench_{LLM}$  and  $Bench_{Text}$ .

### 4.3 Experimental Results

Consider the set of system-generated spans  $S = \{s_1, \dots, s_n\}$  and ground truth spans  $G = \{g_1, \dots, g_m\}$ . We measure the number of textual spans that are correctly identified, labeled, and attributed to the valid supporting evidence as  $ov$ . Then, we calculate verification system’s precision as  $\frac{ov}{|S|}$ , recall as  $\frac{ov}{|G|}$ , and the F1 score. Table 1 shows the performance of our verification systems on both evaluation datasets. As illustrated,  $FLEEK_{GPT-3}$  outperforms  $FLEEK_{Vicuna}$  on both datasets by a large margin (12 F1 pts). However, given that Vicuna is about  $5\times$  smaller than GPT-3, the average performance of  $FLEEK_{Vicuna}$  (71.27 F1) shows its efficiency in Fact Verification. Moreover, the results show that both systems can identify the “Strongly Supported” facts with high precision and recall. However, they fail to detect all facts or attribute them to the correct evidence for “Likely Supported” or “Questionable” cases.

We also measure the accuracy of revisions proposed by the fact correction component. Both systems have on-par performance with an average accuracy of 72.7%. However, our investigation shows that 54.1% of incorrect revisions are a result of errors in previous components propagated through the system. Thus, Fact Correction’s average precision, given the correct verification results, is 87.5%.

Note that although our initial results show great promise, both evaluation datasets are small (50 sentences) and come from the same data source (Wikipedia). One ongoing work is to create a larger benchmark (with different levels of difficulty from more diverse sources) for a more extensive and reliable evaluation of our system.

**Error Analysis.** We randomly select 30 examples where  $FLEEK_{GPT-3}$  made erroneous decisions and investigate the types of errors each of its components made (Figure 5). In general, the Fact Extraction component accounted for a significant portion, approximately 49%, of the total errors. This emphasizes the difficulty of mastering

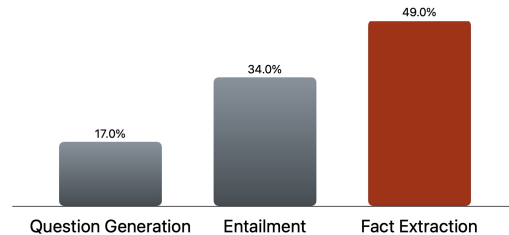


Figure 5: Percentage of total errors generated by different components of  $FLEEK_{GPT-3}$ .

Fact Extraction through in-context learning. Errors produced by this component include, but are not limited to, wrong triple format, broken n-ary relations, missing triples, and hallucination. Figure 5 further indicates that the GPT-3 might not excel in reasoning, as the entailment component also contributes significantly to system errors.

## 5 Conclusion and Future Work

We presented FLEEK, an innovative solution geared towards assisting users in verifying the accuracy and factuality of textual claims. We aim to keep improving the FLEEK so that it can be a handy tool for various stakeholders. As part of our future work, we intend to do more comprehensive evaluations of FLEEK, including testing it with various LLMs and over a comprehensive benchmark.

**Limitation.** First, our current system depends on the initial set of responses generated by LLMs to perform the tasks. Nevertheless, we can prompt each component multiple times and employ methods such as majority voting to enhance the accuracy of each task. Second, experiments presented are based on small-scale datasets. We plan to expand both datasets as part of our future endeavors. Finally, both datasets are manually annotated by one annotator. We plan to hire more annotators and refine the annotation process so as to provide a more comprehensive evaluation of our method.

## References

- Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. [FEVEROUS: Fact extraction and VERification over unstructured and structured information](#).
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%\\* chatgpt quality](#).
- Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. [Knowledge vault: A web-scale approach to probabilistic knowledge fusion](#). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, page 601–610, New York, NY, USA. Association for Computing Machinery.
- Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. [Saga: A platform for continuous construction and serving of knowledge at scale](#). In *Proceedings of the 2022 International Conference on Management of Data*. ACM.
- Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022. [Paragraph-based transformer pre-training for multi-sentence inference](#).
- Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. [Fine-grained fact verification with kernel graph attention network](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online. Association for Computational Linguistics.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jipapu Wang, and Xindong Wu. 2023. [Unifying large language models and knowledge graphs: A roadmap](#).
- Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. 2023. [Knowledge graphs: Opportunities and challenges](#).
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [Stanford alpaca: An instruction-following llama model](#). [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca).
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a large-scale dataset for fact extraction and VERification](#). In *NAACL-HLT*.
- Helena Webb, Marina Jirotko, Bernd Carsten Stahl, William Housley, Adam Edwards, Matthew Williams, Rob Procter, Omer Rana, and Pete Burnap. 2016. [Digital wildfires: Hyper-connectivity, havoc and a global ethos to govern social media](#). *SIGCAS Comput. Soc.*, 45(3):193–201.
- Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. [WikiQA: A challenge dataset for open-domain question answering](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.
- Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2019. [Reasoning over semantic-level graph for fact checking](#). *CoRR*, abs/1909.03745.

# YATO: Yet Another deep learning based Text analysis Open toolkit

Zeqiang Wang<sup>1\*</sup>, Yile Wang<sup>2\*</sup>, Jiageng Wu<sup>1</sup>, Zhiyang Teng<sup>3</sup>, Jie Yang<sup>1†</sup>

<sup>1</sup>Zhejiang University

<sup>2</sup>Institute for AI Industry Research (AIR), Tsinghua University

<sup>3</sup>Nanyang Technological University

zeqiangwang.medicalai@outlook.com wangyile@air.tsinghua.edu.cn

jiagengwu@zju.edu.cn zhiyang.teng@ntu.edu.sg jieynlp@gmail.com

## Abstract

We introduce YATO, an open-source, easy-to-use toolkit for text analysis with deep learning. Different from existing heavily engineered toolkits and platforms, YATO is lightweight and user-friendly for researchers from cross-disciplinary areas. Designed in a hierarchical structure, YATO supports free combinations of three types of widely used features including 1) traditional neural networks (CNN, RNN, *etc.*); 2) pre-trained language models (BERT, RoBERTa, ELECTRA, *etc.*); and 3) user-customized neural features via a simple configurable file. Benefiting from the advantages of flexibility and ease of use, YATO can facilitate fast reproduction and refinement of state-of-the-art NLP models, and promote the cross-disciplinary applications of NLP techniques. The code, examples, and documentation are publicly available at <https://github.com/jiesutd/YATO>. A demo video is also available at <https://www.youtube.com/playlist?list=PLJ0mhZMcRuDU1TkzBfAft0qiJRxYTTjXH>.

## 1 Introduction

Large language models (LLMs) such as GPT-3 (Brown et al., 2020), ChatGPT (OpenAI, 2022), and LLaMA (Touvron et al., 2023a,b) have gained significant progress in natural language processing (NLP), showing strong abilities to understand text and competitive performance across various NLP tasks. However, these models are either close-source or difficult to fine-tune due to the high computational costs, which makes them inconvenient for academic research or practical implementation.

Alternatively, traditional neural models, such as recurrent neural networks (RNN, Hochreiter and Schmidhuber, 1997), convolutional neural networks (CNN, LeCun et al., 1989), and pre-trained language models (PLMs, Devlin et al., 2019; Liu

et al., 2019; Clark et al., 2020) have been widely studied and utilized for text understanding. These models benefit from large-scale training data and can be quickly fine-tuned toward specific usages. Recent works also show they can offer useful guidance to LLMs (Xu et al., 2023). Therefore, small open-source deep learning models are important in current NLP systems, especially in computation and data resource-limited scenarios.

However, due to the complexity of the deep learning model architecture, it is challenging to implement methods or reproduce results from the literature. The different implementations of these models can lead to unfair comparisons or misleading results. Most existing frameworks were designed for professional developers, which brings additional obstacles for less experienced users, especially for researchers with less or no artificial intelligence (AI) background (Zacharias et al., 2018; Zhang et al., 2020; Johnson et al., 2021). In addition, these frameworks seldom support user-defined features required for various domain applications (*e.g.*, in medical named entity recognition, customized lexicons can be supplemented as external features, such that additional labels are tagged as features when a word occurs in the lexicon). For non-expert, cross-domain users, customizing models via source code with additional features is complex. To promote interdisciplinary applications of cutting-edge NLP techniques, it is necessary to build a flexible, user-friendly, and effective text representation framework that supports a wide range of deep learning architectures and customized domain features.

There exist several text analysis toolkits in the NLP community. CoreNLP (Manning et al., 2014) and spaCy (Honnibal and Montani, 2017) offer pipelines for many traditional NLP tasks, while the performance is sometimes under-optimal due to the use of less powerful models. AllenNLP (Gardner et al., 2017) and flairNLP (Akbik et al., 2019) utilize pre-trained models while they do not support

\*Equal contribution.

†Corresponding author.

| System       | Lightweight | PLM | Neural Models | User-Defined Features | Configurable w/o Coding | SOTA Performance | Reference                    |
|--------------|-------------|-----|---------------|-----------------------|-------------------------|------------------|------------------------------|
| CoreNLP      | ✓           | ✗   | ✗             | ✗                     | ✗                       | ✗                | (Manning et al., 2014)       |
| spaCy        | ✓           | ✓   | ✓             | ✗                     | ✗                       | ✗                | (Honnibal and Montani, 2017) |
| AllenNLP     | ✓           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Gardner et al., 2017)       |
| FlairNLP     | ✓           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Akbik et al., 2019)         |
| NCRF++       | ✓           | ✗   | ✓             | ✓                     | ✓                       | ✓                | (Yang and Zhang, 2018)       |
| FairSeq      | ✗           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Ott et al., 2019)           |
| Transformers | ✗           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Wolf et al., 2020)          |
| PaddleNLP    | ✗           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Contributors, 2021)         |
| EasyNLP      | ✗           | ✓   | ✓             | ✗                     | ✗                       | ✓                | (Wang et al., 2022a)         |
| <b>YATO</b>  | ✓           | ✓   | ✓             | ✓                     | ✓                       | ✓                | <b>This paper</b>            |

Table 1: Comparison between existing popular text analysis libraries and our proposed YATO.

user-defined features. FairSeq (Ott et al., 2019) is designed for sequence-to-sequence tasks like machine translation and document summarization. Transformers (Wolf et al., 2020) offers implementation for various tasks by using state-of-the-art models across different modalities, while it is heavily engineered. PaddleNLP (Contributors, 2021) and EasyNLP (Wang et al., 2022a) are specifically designed for industrial application and commercial usage, which are not lightweight for research purposes. The above toolkits are mostly developed for professional AI researchers or engineers, where heavy coding effort is necessary during model development and deployment. The learning curve is steep to fully leverage these toolkits for cross-disciplinary researchers (*e.g.*, medical, financial) who need to build models with lightweight code.

This paper presents a toolkit, **YATO** (Yet Another deep learning based Text analysis Open toolkit), for researchers looking for a convenient way of building state-of-the-art models for two most popular types of NLP tasks: sequence labeling (*e.g.*, Part-of-Speech tagging, named entity recognition) and sequence classification (*e.g.*, sentiment analysis, document classification). YATO is built on NCRF++ (Yang and Zhang, 2018), a popular neural sequence labeling toolkit with over 250+ citations from research papers, 1,900+ stars and 120+ merged pull requests on GitHub as of Oct. 2023. NCRF++ has been utilized in many cross-disciplinary research projects, including medical (Yang et al., 2020) and finance (Wan et al., 2021). YATO retains its strengths, integrates advanced pre-trained language models, and adds capabilities for sequence classification and data visualization.

## 2 Highlights of YATO

Table 1 lists the comparison of YATO and popular existing text analysis libraries. The highlights of YATO include:

- **Lightweight.** YATO focuses on two fundamental while popular NLP tasks: sequence labeling and sequence classification, covering many downstream applications such as information extraction, sentiment analysis, text classification, *etc.* Different from the heavily engineered libraries, YATO is concise and lightweight with less library dependence. It can be fast developed and deployed in various environments, making it a user-friendly toolkit for less experienced users.

- **Flexible.** Most of the existing libraries do not support the combination of various neural features. By using YATO, users can customize their models through free combinations of various neural models, including traditional neural networks (CNN, RNN) and state-of-the-art PLMs, as well as hand-crafted features for domain adaptation. YATO also supports various inference layers, including attention pooling, softmax, conditional random field (CRF), and *nbest* decoding.

- **Configurable.** To minimize the effort of coding, all the model developments on YATO can be easily conducted by editing the configuration file. YATO will load the configuration file and construct the deep learning models following the configurations.

- **Easy to Use.** YATO is built based on PyTorch<sup>1</sup> and has been released on PyPI<sup>2</sup>, the installation can be done through `pip install ylab-yato`. For non-AI users, editing a configuration file to build deep learning models is simple and intuitive. For AI users, YATO provides various modularized functions for professional development.

- **High Performance.** In extensive experiments on sequence labeling and classification tasks, YATO proves that it can achieve state-of-the-art performance on most tasks and datasets. YATO offers flexibility in terms of hardware resources, supporting both GPU and CPU for training and inference

<sup>1</sup><https://pytorch.org/>.

<sup>2</sup><https://pypi.org/>.



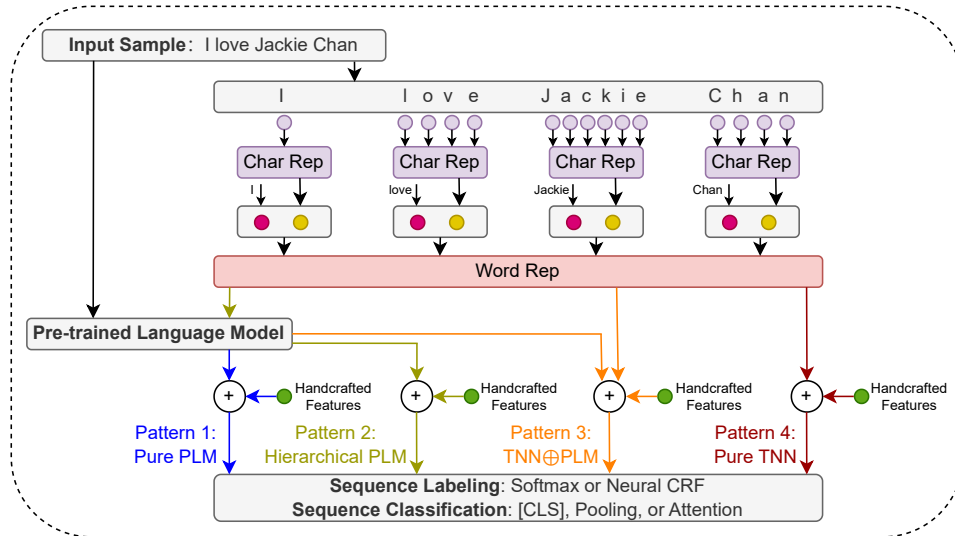


Figure 1: The overall architecture of YATO. The purple, red, yellow, and green circles represent character embeddings, word embeddings, character sequence representation, and custom handcrafted features, respectively.

processes. It provides the ability to specify the desired device configuration, facilitating efficient utilization of multiple GPUs on a single server.

- **Visualization.** YATO offers the interface for visualizing text attention, which can help users further interpret and analyze the results.

### 3 Architecture of YATO

YATO is designed *hierarchically* to support flexible combinations of character-level features, word-level features, and pre-trained language models<sup>3</sup>, as well as handcrafted features. As illustrated in Figure 1, YATO supports four patterns to represent text as embeddings and with flexible choices on adding handcrafted features and inference layers.

#### 3.1 Text Representations

**Pure Pre-trained Language Model.** YATO enables the initialization of parameters with pre-trained language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ELECTRA (Clark et al., 2020), and fine-tunes them on training data. Leveraging the rich knowledge inside the PLM, they have demonstrated strong performance on downstream tasks. To better leverage the models with domain-specific knowledge, YATO also supports pre-trained models designed for specific tasks, such as SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2020) and others.

**Hierarchical Pre-trained Language Model.** The

hierarchical pre-trained language model in YATO differs from the conventional notion of hierarchy, which typically describes relationships between word, sentence, and document structures. Instead, it signifies the ordinal relation between the traditional neural network and the pre-trained language model. Specifically, YATO supports using both word sequence features and the pre-trained language model representations in a hierarchical way, where the word and character features can be explicitly encoded in advance and used as the input for the pre-trained language model.

**Traditional Neural Network (TNN) & Pre-trained Language Model.** In contrast to the hierarchical combination, we can use the word sequence features directly before the final prediction layer, combined with the representation brought from the PLMs. Such a feature-based approach is also used in ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), which shows close performance while does not require fine-tuning the pre-trained models.

**Pure Traditional Neural Network.** Besides using the transformer-based pre-trained models, we also support traditional neural models such as RNN, CNN, and BiLSTM. Compared with Transformer, these models usually have fewer parameters and are also shown effective for sequence modeling (Ma and Hovy, 2016; Lample et al., 2016; Yang et al., 2018), especially when the training data is limited.

#### 3.2 Handcrafted Features and Inference

**Handcrafted Features .** YATO provides feature embedding modules to encode any handcrafted fea-

<sup>3</sup>It supports all available pre-trained models from <https://huggingface.co/models>.

---

```

Configuration File

#Dataloader
train_dir=The path of train dataset
dev_dir=The path of development dataset
...
#Model
use_crf=True/False
use_char=True/False
char_seq_feature=GRU/LSTM/CNN/False
word_seq_feature=GRU/LSTM/CNN/FeedFowrd/False
low_level_transformer=pretrain language model
high_level_transformer=PLM from huggingface
bilstm=True/False
...
#Handcrafted Features
feature=[POS] emb_size=20 emb_dir=your POS embedding
feature=[Cap] emb_size=20 emb_dir=your Cap embedding
...
#Hyperparameters
sentence_classification=True/False
status=train/decode
iteration=epoch number
batch_size=batch size
optimizer=SGD/Adagrad/adadelta/rmsprop/adam/adamw
learning_rate=learning rate
...
#Prediction
raw_dir=The path of decode file
decode_dir=The path of the decode result file
nbest=0 (for labeling)/1 (for classification)
...

```

---

Table 2: A sample of a configuration file.

tures and the corresponding feature embeddings can be concatenated to the text representation on any patterns. Users can add feature embeddings by formatting the text following YATO instructions without any coding effort.

**Inference Layers.** The encoded text embeddings from the text representation structure are fed into a softmax or CRF (Lafferty et al., 2001) layer for sequence labeling tasks, and YATO also supports *nbest* decoding to generate more candidate label sequences with probabilities for further optimization. For sequence classification tasks, a classifier head over [CLS] representation, or pooling/attention layer on all word representations is used, and a visualization tool is also available in YATO to visualize the word importance in the attention layer.

## 4 The Overall Workflow and Usage

The overall workflow of YATO comprises three primary components to facilitate solving downstream tasks: configuration preparation, model training, and model decoding. Users can leverage this modular pipeline to tackle a wide range of applications. **Configuration Preparation.** Users can specify the

dataset, model, optimizer, and decoding through the configuration file, as shown in Table 2. In particular, different patterns can be customized by setting the values of “high\_level\_transformer”, “low\_level\_transformer”, “bilstm”, and different word sequence representations can be easily designed through specifying “char\_seq\_feature” and “word\_seq\_feature”. Handcrafted features can be added through “feature”.

**Model Training.** YATO enables efficient training of high-performance models for sequence labeling and sentence classification with minimal code and configuration file specification. For example, users can train a competitive named entity recognition or text classification model using a few lines of code and a configuration file. The simplicity of the YATO interface allows rapid prototyping and experimentation for these fundamental NLP tasks. The framework was designed in batch computation which can fully utilize the power of GPUs.

**Model Decoding.** Similar to model training, simple file configuration can be used to enable YATO. Besides the greedy decoding, YATO also supports *nbest* decoding, *i.e.*, which decodes label sequences with the top *n* probabilities by using the Viterbi decoding in neural CRF layers. The *nbest* results can serve as important resources for further optimizations, *e.g.*, reranking (Yang et al., 2017).

## 5 Experiments

### 5.1 Datasets and Main Results

To evaluate our framework, we evaluated 8 datasets that cover sequence labeling and classification tasks in both English and Chinese, including named entity recognition (NER) on CoNLL2003 (Tjong Kim Sang and De Meulder, 2003), OntoNotes (Hovy et al., 2006) and MSRA (Levow, 2006); CCG supertagging on CCG-Bank (Hockenmaier and Steedman, 2007); sentiment analysis on SST2, SST5 (Socher et al., 2013), and ChnSentiCorp (Tan and Zhang, 2008).

Table 3 and Table 4 demonstrate that YATO can reproduce both classical and state-of-the-art deep learning models on most sequence labeling and classification tasks. For some results such as BERT on CoNLL, the originally reported 92.4 F1 score by Devlin et al. (2019) may not be achieved with current libraries, as discussed in previous literature (Stanislawek et al., 2019; Gui et al., 2020). Overall, YATO achieves the best performance on MSRA, OntoNotes 4.0, CCG supertagging, and

| Model          | CoNLL 2003   |                    | OntoNotes 5.0 |                    | MSRA         |                    | Ontonotes 4.0 |                    | CCG supertagging |                    |
|----------------|--------------|--------------------|---------------|--------------------|--------------|--------------------|---------------|--------------------|------------------|--------------------|
|                | YATO         | Ref.               | YATO          | Ref.               | YATO         | Ref.               | YATO          | Ref.               | YATO             | Ref.               |
| CCNN+WLSTM+CRF | <b>91.26</b> | 91.11 <sup>a</sup> | 81.53         | -                  | 92.83        | -                  | 74.55         | -                  | 93.80            | -                  |
| BERT-base      | 91.61        | 92.4 <sup>b</sup>  | 84.68         | 85.54 <sup>e</sup> | <b>95.81</b> | 94.71 <sup>f</sup> | <b>80.57</b>  | 79.93 <sup>f</sup> | <b>96.14</b>     | 92.16 <sup>h</sup> |
| RoBERTa-base   | <b>90.23</b> | 90.11 <sup>c</sup> | <b>86.28</b>  | 86.2 <sup>c</sup>  | <b>96.02</b> | 95.1 <sup>g</sup>  | <b>80.94</b>  | 80.37 <sup>e</sup> | 96.16            | -                  |
| ELECTRA-base   | <b>91.59</b> | 91.5 <sup>d</sup>  | 85.25         | 87.6 <sup>d</sup>  | 96.03        | -                  | 80.47         | -                  | 96.29            | -                  |

Table 3: Results for sequence labeling tasks. **Bold** represents that YATO’s re-produce is more accurate. <sup>a</sup>(Yang et al., 2018) <sup>b</sup>(Devlin et al., 2019) <sup>c</sup>(Liang et al., 2020) <sup>d</sup>(Shelmanov et al., 2021) <sup>e</sup>(Wang et al., 2022b) <sup>f</sup>(Liu et al., 2021) <sup>g</sup>(Li et al., 2022) <sup>h</sup>(Durrani et al., 2020)

| Model          | SST2                      |                    | SST5         |                   | ChnSentiCorp |                   |
|----------------|---------------------------|--------------------|--------------|-------------------|--------------|-------------------|
|                | YATO                      | Ref.               | YATO         | Ref.              | YATO         | Ref.              |
| CCNN+WLSTM+CRF | 87.61                     | -                  | 43.48        | -                 | 88.22        | -                 |
| BERT-base      | <b>93.00</b> <sup>†</sup> | 92.7 <sup>†a</sup> | <b>53.48</b> | 53.2 <sup>d</sup> | <b>95.96</b> | 95.3 <sup>g</sup> |
| RoBERTa-base   | 92.55 <sup>†</sup>        | 94.8 <sup>†b</sup> | 51.99        | 56.4 <sup>e</sup> | <b>96.04</b> | 95.6 <sup>h</sup> |
| ELECTRA-base   | 94.72 <sup>†</sup>        | 95.1 <sup>†c</sup> | <b>55.11</b> | 54.8 <sup>f</sup> | <b>95.96</b> | 94.5 <sup>g</sup> |

Table 4: Results of sequence classification tasks. **Bold** represents that YATO re-produce is more accurate. † denotes the results of the dev set. <sup>a</sup>(Devlin et al., 2019) <sup>b</sup>(Zaheer et al., 2020) <sup>c</sup>(Clark et al., 2020) <sup>d</sup>(Munikaer et al., 2019) <sup>e</sup>(Sun et al., 2020) <sup>f</sup>(Xia et al., 2022) <sup>g</sup>(Cui et al., 2021) <sup>h</sup>(Xin et al., 2020)

| Patterns            | SST5         | CoNLL 2003   |
|---------------------|--------------|--------------|
| 1. Pure PLM         | 53.48        | <b>91.61</b> |
| 2. Hierarchical PLM | 53.77        | 90.52        |
| 3. TNN $\oplus$ PLM | <b>54.84</b> | 90.47        |
| 4. Pure TNN         | 43.48        | 91.26        |

Table 5: Performances of different training patterns.

ChnSentiCorp. The compatibility and reproducibility across different models and tasks demonstrate that YATO can serve as a platform for reproducing and comparing different methods from classical neural models to state-of-the-art PLMs.

## 5.2 Comparison of Different Patterns

Table 5 shows the performance of four different model patterns on both sequence labeling and classification tasks (one dataset for each task). The combination of Hierarchical PLM and TNN $\oplus$ PLM (patterns 2 and 3) outperforms pure models (patterns 1 and 4) on SST5. However, pure PLM achieves the best performance on the CoNLL 2003 NER dataset. These results demonstrate that complex models are not always better than simple models, and a flexible framework is necessary for providing various model candidates.

## 5.3 Results by Using Handcraft Features

To demonstrate the effectiveness of encoding handcraft features in domain application, Table 6 shows

|                       | NCBI-disease       | Yidu-s4k                |
|-----------------------|--------------------|-------------------------|
| Pure PLM              | 84.23              | 82.73                   |
| +Handcrafted Features | <b>84.85</b> (Cap) | 82.98 (Lexicon2)        |
|                       | 84.78 (Lexicon1)   | <b>83.46</b> (Lexicon3) |

Table 6: Handcraft features. Lexicon1 (English medical glossary), Lexicon2 (medical word list from THUOCL), and Lexicon3 (Chinese medical glossary).

the comparison results on two medical NER tasks, the NCBI-disease (Doğan et al., 2014) for English and Yidu-S4K (CCKS, 2019) for Chinese. Experiments on NCBI-disease apply two types of features, capitalization and English medical lexicon from the Chinese-English mapping medical glossary<sup>4</sup>. Experiments on Yidu-s4k dataset employ two medical lexicons as handcrafted features: the medical glossary of THUOCL (Han et al., 2016) (THU Open Chinese Lexicon) and the same medical glossary sourced from the web<sup>4</sup> while in Chinese version. Results show that handcraft features can improve the model performance in the medical domain.

## 5.4 Comparison with Transformers

The aforementioned results show that we can achieve the reported values across various tasks by using YATO. We further use tasks from GLUE benchmark (Wang et al., 2018) and compare with the results by using Huggingface Transform-

<sup>4</sup><http://medtop10.com>

Negative **dreary** **tip-off** **of** **goodfellas** **that** **erves** **as** **a** **muddled** **and** **offensive** cautionary tale for hispanic americans .

Negative nothing **debases** **a** **concept** **comedy** **quite** **like** **the** **grinding** **of** **bad** **ideas** , **and** **showtime** **is** **prammed** **into** **them** .

Positive it 's **one** **of** **the** **saddest** **films** **i** **have** **ever** **seen** **that** **still** **manages** **to** **be** **uplifting** **but** **not** **overly** **sentimental** .

Positive **a** **fairly** **by-the-books** **blend** **of** **action** **and** **romance** **with** **sprinklings** **of** **intentional** **and** **unintentional** **comedy** .

Negative 送包挺贵的，做工也不好，商家上99块钱，款式芝  
笔记本包挺贵的，商家送好的包。

Negative 这旅游热城市但酒店贵的地方，酒店在深夜--  
因为咱队其他地方是不热。

Positive 画质细腻触感板凸点处理，手  
感也不错白色键盘亮眼。

Positive 物美的好选择，晚上到机住，服务热情，  
没地毯，海景不错。

Figure 2: Visualization of attention weights. Different degrees of background color reflect the distributions of attention in words or characters.

| Tasks (Metric)                 | Transformers | YATO         |
|--------------------------------|--------------|--------------|
| MRPC (Accuracy)                | 84.07        | <b>85.78</b> |
| QQP (Accuracy)                 | 90.71        | <b>90.81</b> |
| MNLI (Matched Accuracy)        | 83.91        | <b>84.21</b> |
| QNLI (Accuracy)                | <b>90.66</b> | 90.28        |
| SST2 (Accuracy)                | 92.32        | <b>93.00</b> |
| RTE (Accuracy)                 | <b>65.70</b> | 63.90        |
| CoLA (Correlation Coefficient) | 56.53        | <b>58.00</b> |
| <b>Average</b>                 | 80.55        | <b>80.85</b> |

Table 7: Results of fine-tuning BERT-base-uncased model on GLUE benchmark.

ers (Wolf et al., 2020), which is one of the most popular libraries. Table 7 shows the results by using BERT-base-uncased model, the values of Huggingface Transformers are sourced from the corresponding github page<sup>5</sup>. YATO achieves comparable and overall better performance than that of Huggingface Transformers by using default settings.

### 5.5 Visualization of Attention Map

Beyond performance, YATO provides a visualization tool for taking the list of words and the corresponding weights as input to generate Latex code for visualizing the attention-based result. Figure 2 provides visualization examples of attention on sentiment prediction tasks. Words or characters with sentiment polarities can be automatically extracted and highlighted using our YATO module. As shown in this table, words that have a high impact on the sentiment are highlighted. This visualization module can improve the interpretability of deep learning models in our toolkit.

### 5.6 Efficiency Analysis

YATO is implemented using a fully batch computing approach, making it quite efficient in both model training and decoding. With the help of GPU and large batches, models built on YATO can be decoded efficiently. Figure 3 shows the

<sup>5</sup><https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification>

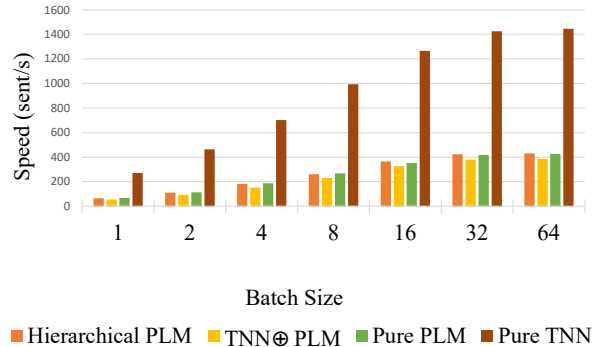


Figure 3: Decoding speed for the four patterns on different batch sizes. Tested on NVIDIA RTX 2080Ti GPU.

decoding speed of four patterns at different batch sizes. The Pure TNN model has the best inference speed (1400 sentences/s) with a batch size of 64. The decoding speed of Transformer-based models decreases to around 400 sentences/s, showing the trade-off between efficiency and performance. Overall, without using external optimization techniques, YATO has a competitive decoding speed.

## 6 Conclusion

YATO is an open-source toolkit for text analysis that supports various combinations of state-of-the-art deep learning models and user-customed features, with high flexibility and minimum coding effort. YATO is maintained by core developers from YLab (<https://ylab.top/>). It aims to help AI researchers build state-of-the-art NLP models and assist non-AI researchers in conducting cross-disciplinary research with advanced NLP techniques. Given the success of its predecessor, NCRF++, we believe that YATO will greatly promote the applications of NLP in various cross-disciplinary fields and reduce disparities of AI application in these areas. In the future, we plan to integrate advanced LLMs and customize modules that support modeling time series, multimodal features, and specific features for various domains.

## Limitations

Our proposed text analysis toolkit mainly focuses on discriminative style tasks, where most of them are treated as token-level or sentence-level classification tasks. Recent studies show that the generative style language models such as GPT (Radford et al., 2018), BART (Lewis et al., 2019), and T5 (Raffel et al., 2020) can also show promising zero-shot and few-shot results by adding user-defined prompts or instructions as external inputs, we leave this as our future work.

## 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 *Proc. of NAACL, (Demonstrations)*.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. *Scibert: Pretrained language model for scientific text*. In *EMNLP*.
- 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.
- CCKS. 2019. China conference on knowledge graph and semantic computing 2019 named entity recognition of chinese electronic medical record. <https://www.sigkg.cn/ccks2019/en/>.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proc. of ICLR*.
- PaddleNLP Contributors. 2021. Paddlenlp: An easy-to-use and high performance nlp library. <https://github.com/PaddlePaddle/PaddleNLP>.
- Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for Chinese BERT. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3504–3514.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.
- Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.
- Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proc. of EMNLP*.
- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. *Allennlp: A deep semantic natural language processing platform*.
- Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. 2020. *Uncertainty-aware label refinement for sequence labeling*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2316–2326, Online. Association for Computational Linguistics.
- Shiyi Han, Yuhui Zhang, Yunshan Ma, Cunchao Tu, Zhipeng Guo, Zhiyuan Liu, and Maosong Sun. 2016. *Thuocl: Tsinghua open chinese lexicon*. *Tsinghua University*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.
- Matthew Honnibal and Ines Montani. 2017. *spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing*.
- Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. *OntoNotes: The 90% solution*. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.
- Kyle P Johnson, Patrick J Burns, John Stewart, Todd Cook, Clément Besnier, and William JB Mattingly. 2021. The classical language toolkit: An nlp framework for pre-modern languages. 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 20–29.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. [Back-propagation applied to handwritten zip code recognition](#). *Neural Computation*, 1(4):541–551.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Gina-Anne Levow. 2006. [The third international Chinese language processing bakeoff: Word segmentation and named entity recognition](#). In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia. 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*.
- Linyang Li, Yong Dai, Duyu Tang, Zhangyin Feng, Cong Zhou, Xipeng Qiu, Zenglin Xu, and Shuming Shi. 2022. Markbert: Marking word boundaries improves chinese bert. *arXiv preprint arXiv:2203.06378*.
- Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proc. of SIGKDD*.
- Wei Liu, Xiyan Fu, Yue Zhang, and Wenming Xiao. 2021. [Lexicon enhanced Chinese sequence labeling using BERT adapter](#). 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 5847–5858, 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 e-prints*, page arXiv:1907.11692.
- Xuezhe Ma and Eduard Hovy. 2016. [End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL: System Demonstrations*.
- Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE.
- OpenAI. 2022. Chatgpt: <https://chat.openai.com/>.
- 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 *Proc. of NAACL-HLT: Demonstrations*.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.
- 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Artem Shelmanov, Dmitri Puzyrev, Lyubov Kupriyanova, Denis Belyakov, Daniil Larionov, Nikita Khromov, Olga Kozlova, Ekaterina Artemova, Dmitry Dylov, and Alexander Panchenko. 2021. [Active learning for sequence tagging with deep pre-trained models and bayesian uncertainty estimates](#). pages 1698–1712.
- 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 *Proc. of EMNLP*.
- Tomasz Stanislawek, Anna Wróblewska, Alicja Wójcicka, Daniel Ziembicki, and Przemyslaw Biecek. 2019. [Named entity recognition - is there a glass ceiling?](#) In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 624–633, Hong Kong, China. Association for Computational Linguistics.
- Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Self-explaining structures improve nlp models. *arXiv preprint arXiv:2012.01786*.

- Songbo Tan and Jin Zhang. 2008. An empirical study of sentiment analysis for chinese documents. *Expert Systems with Applications*, 34(4):2622–2629.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of NAACL*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shrutu Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Xingchen Wan, Jie Yang, Slavi Marinov, Jan-Peter Calleys, Stefan Zohren, and Xiaowen Dong. 2021. Sentiment correlation in financial news networks and associated market movements. *Scientific reports*, 11(1):3062.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.
- Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022a. [Easynlp: A comprehensive and easy-to-use toolkit for natural language processing](#).
- Shuhe Wang, Xiaoya Li, Yuxian Meng, Tianwei Zhang, Rongbin Ouyang, Jiwei Li, and Guoyin Wang. 2022b. *knn-ner*: Named entity recognition with nearest neighbor search. *arXiv preprint arXiv:2203.17103*.
- 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 *Proc. of EMNLP: System Demonstrations*.
- Mengzhou Xia, Mikel Artetxe, Jingfei Du, Danqi Chen, and Ves Stoyanov. 2022. Prompting electra: Few-shot learning with discriminative pre-trained models. *arXiv preprint arXiv:2205.15223*.
- Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. [DeeBERT: Dynamic early exiting for accelerating BERT inference](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.
- Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, and Julian McAuley. 2023. Small models are valuable plug-ins for large language models. *arXiv preprint arXiv:2305.08848*.
- Jie Yang, Shuailong Liang, and Yue Zhang. 2018. [Design challenges and misconceptions in neural sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Jie Yang, Liqin Wang, Neelam A Phadke, Paige G Wickner, Christian M Mancini, Kimberly G Blumenthal, and Li Zhou. 2020. Development and validation of a deep learning model for detection of allergic reactions using safety event reports across hospitals. *JAMA Network Open*, 3(11):e2022836–e2022836.
- Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proc. of ACL, System Demonstrations*.
- Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 784–792.
- Jan Zacharias, Michael Barz, and Daniel Sonntag. 2018. A survey on deep learning toolkits and libraries for intelligent user interfaces. *arXiv preprint arXiv:1803.04818*.
- Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in NeurIPS*.
- Yue Zhang, Yile Wang, and Jie Yang. 2020. Lattice lstm for chinese sentence representation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1506–1519.

# Spacerini: Plug-and-play Search Engines with Pyserini and Hugging Face

Christopher Akiki<sup>1,2</sup>

Odunayo Ogundepo<sup>3</sup>

Aleksandra Piktus<sup>4,5</sup>

Xinyu Zhang<sup>3</sup>

Akintunde Oladipo<sup>3</sup>

Jimmy Lin<sup>3</sup>

Martin Potthast<sup>1,2</sup>

<sup>1</sup> Leipzig University    <sup>2</sup> ScaDS.AI    <sup>3</sup> University of Waterloo  
<sup>4</sup> Hugging Face    <sup>5</sup> Sapienza University

## Abstract

We present Spacerini, a tool that integrates the Pyserini toolkit for reproducible information retrieval research with Hugging Face to enable the seamless construction and deployment of interactive search engines. Spacerini makes state-of-the-art sparse and dense retrieval models more accessible to non-IR practitioners while minimizing deployment effort. This is useful for NLP researchers who want to better understand and validate their research by performing qualitative analyses of training corpora, for IR researchers who want to demonstrate new retrieval models integrated into the growing Pyserini ecosystem, and for third parties reproducing the work of other researchers. Spacerini is open-source<sup>1</sup> and includes utilities for loading, preprocessing, indexing, and deploying search engines locally and remotely. We demonstrate a portfolio of 13 search engines created with Spacerini for different use cases.<sup>2</sup>

## 1 Introduction

The commoditization of data has transformed data-driven computer science in general (Hey et al., 2009) and machine learning (ML) and natural language processing (NLP) in particular (Mitchell et al., 2022). The race to train ever-larger language models depends so much on having access to immense amounts of text (Hoffmann et al., 2022) that the datasets have become, as Bender et al. (2021) claims, both technically (Bender et al., 2021) and methodologically (Jo and Gebru, 2020) “too big to document”. In practice, this often leads to an approach of training first and asking questions later (Akiki et al., 2022), which again is an example of a convenience experiment (Krohs, 2012), a research approach that depends on the availability of a resource and the ease of a method rather than its suitability to the problem at hand. In this sense, Beaulieu and Leonelli (2021) believe it is important to distinguish between the availability of data

and its appropriateness, especially in light of the misconception that web data represents all human experience and is immune to the ever-widening *digital divide* (Leonelli, 2020). They suggest that the divide not only exists, but limits the representativeness of the web, which in turn reinforces the biases of the artifacts that use it (Bender et al., 2021).

Being unable to easily audit large datasets incentivizes researchers to release models trained on data they do not truly understand (Mitchell et al., 2022) leading to model behaviors that are hard to study, predict or trace (Mitchell et al., 2022; Sidiqui et al., 2022; Akyurek et al., 2022). This is especially problematic in light of the potential real-world harms that ensue (Weidinger et al., 2021; Hutchinson et al., 2020; Founta et al., 2018; Fast et al., 2016). Being able to properly understand the limitations of our datasets and qualitatively explore them in an ad-hoc fashion is a necessary first step toward understanding the behavior, harmful biases and failure modes of the artifacts that build upon them. Understanding the training data is therefore a critical step in the process of releasing and auditing large language models (Mökander et al., 2023).

It is from this vantage point that we initially developed Spacerini as an open-source tool for the quick indexing and deployment of shareable search engines, but have also since come to realize its potential in being useful for an even wider audience interested in making their text artifacts searchable. Indeed this includes IR students, Digital Humanists, Shared Tasks organizers, and digital investigative journalists, all of whom seek to quickly deploy ad hoc search engines for their research. We cover these use cases in more detail in Section 4.

Spacerini helps streamline the process of auditing large datasets by allowing users to effortlessly index their text collections and deploy them as interactive search applications that can be easily edited in the browser and shared with all stakeholders. It achieves that by “standing on the shoul-

<sup>1</sup><https://github.com/castorini/hf-spacerini>

<sup>2</sup><https://hf.co/spacerini>



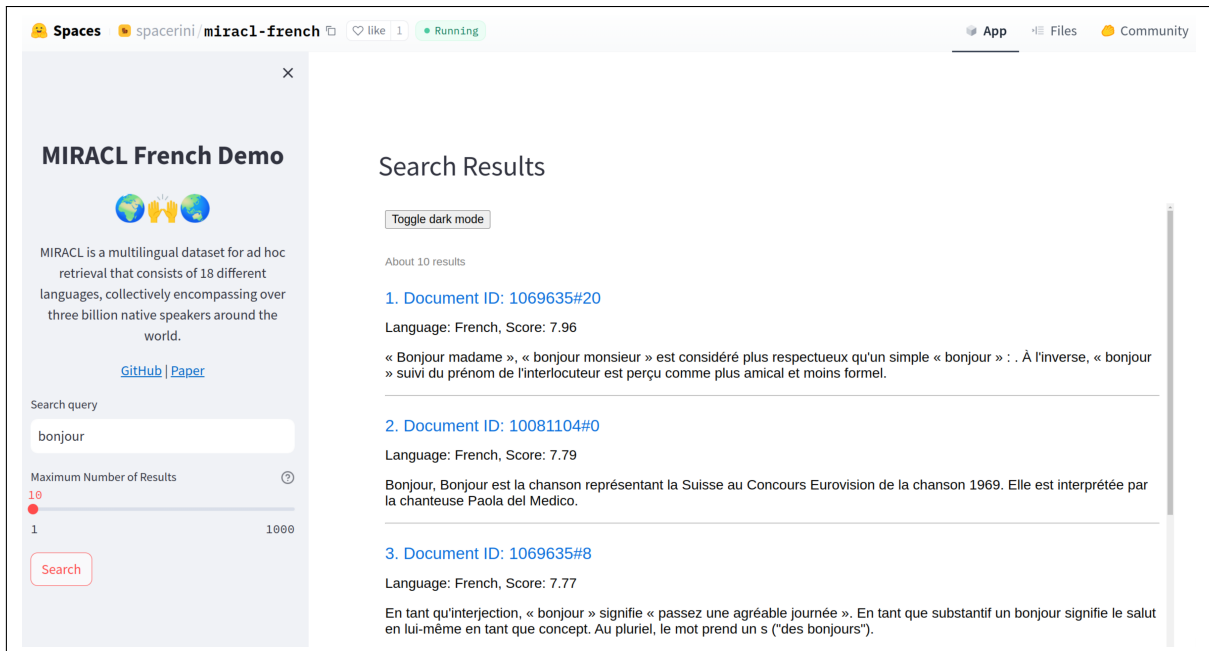


Figure 1: Example of one of the many search apps ([hf/spacerini/miracl-french](https://hf.com/spacerini/miracl-french)) deployed as a Hugging Face Space. The Lucene BM25 index is hosted in the same repository as the frontend using `git LFS` and the frontend is based on a template which was automatically generated from one of the many Spacerini cookiecutter templates. The loading, preprocessing, indexing, and app deployment were made using an end-to-end workflow similar to the one showcased in Section 3.

ders” of battle-tested open-source libraries from the Castorini (Lin et al., 2021), Hugging Face (Abid et al., 2019; Lhoest et al., 2021) and Python ecosystems. It also enhances the interoperability between them to enable quick indexing and free deployment of search interfaces in an easy-to-use package that makes it possible to reduce the overhead typically involved in operationalizing data governance frameworks, and allows stakeholders to focus on data analysis rather than data engineering. This is achieved through the modularity of its design that enables data loading (Section 3.1), preprocessing (Section 3.2), dense and sparse indexing (Sections 3.3), as well as the creation (Section 3.1), and free hosting of graphical search interfaces (Section 3.5) for text datasets.

## 2 Background and Related Work

Large scale, predominantly web-mined text datasets have been proliferating in NLP recently, giving rise to publications (Laurençon et al., 2022; Gao et al., 2021; Ortiz Suárez et al., 2019; Raffel et al., 2022) which often contain interesting analyses of the specific datasets being presented, however, usually lack any comparison to existing resources beyond basic metrics such as sizes of the datasets or languages they contain.

As discussed in Section 1, in the face of an increased scrutiny of the models trained on datasets in question, the topic of data understanding and governance has been gaining more traction, being accepted as an important part of research. Efforts such as those of Mitchell et al. (2022) contribute frameworks for more standardised and reproducible metrics and measurements of datasets, and we position ourselves as a complementary continuation of their work, focusing on a more curatorial and qualitative assessment that might not readily fit under the umbrella of “measurements”. We therefore aim to fill the gap in the evaluation landscape by facilitating qualitative, rather than quantitative analysis of large scale data collections.

Similarly to the authors of Gradio (Abid et al., 2019), a Python package for fast development of Machine Learning demos, we believe that the accessibility of data and model analysis tools is crucial to building both the understanding of and the trust in the underlying resources. The potential of relevance-based interfaces to massive textual corpora, the creation of which can be facilitated by leveraging toolkits such as Pyserini (Lin et al., 2021), has previously been tapped into by the researchers at the Allen Institute of AI who propose

a C4 (Raffel et al., 2022) search engine<sup>3</sup>. Similar interfaces have also been found useful in more specialised domains, e.g. in COVID-related datasets (Zhang et al., 2020), news quotes (Vuković et al., 2022), or medical literature (Niezni et al., 2022). However, while these solutions are undeniably useful, they remain very contextual: dataset-specific, and project-specific. We believe Spacerini to be the first generalizable tool which proposes an end-to-end pipeline automating the route from raw text to qualitative analysis.

### 3 Spacerini

Spacerini is a modular framework that integrates Pyserini with the Hugging Face ecosystem to streamline the process of going from any Hugging Face text dataset (or indeed any text dataset)—either local or hosted on the Hugging Face Hub—to a search interface driven by a Pyserini index that can be deployed for free on the Hugging Face Hub. In what follows, we deconstruct an example script<sup>4</sup> to showcase the different features enabled by Spacerini. When run end-to-end, the script pulls a dataset from Hugging Face, pre-processes it, indexes it, creates a gradio-based search interface and deploys that as a Hugging Face Spaces demo. This is only meant as a feature-complete demo, and we don't expect most people to want to integrate every step into their workflows, but rather to cherry-pick and decide what best to use depending on context.

#### 3.1 Loading Data

All our workflows are backed by the Hugging Face datasets library (Lhoest et al., 2021), itself based on the extremely efficient Apache Arrow format. Datasets is a mature library which provides a standardized interface to any tabular dataset, in particular, to tens of thousands of community datasets hosted on the Hugging Face Hub<sup>5</sup>. The datasets library gives fine-grained control over the lifecycle of tabular datasets, which we choose to abstract away through a set of opinionated data loading functions that cover the use cases we deem relevant to information retrieval. We also add new functionality, such as the ability to load any document dataset from the `ir_datasets` library using for example the following one-liner to load

<sup>3</sup><https://c4-search.apps.allenai.org/>

<sup>4</sup><https://github.com/castorini/hf-spacerini/blob/main/examples/scripts/gradio-demo.py>

<sup>5</sup><https://hf.co/datasets>

MS MARCO (Nguyen et al., 2016) as a Hugging Face datasets.Dataset object using a function from the data subpackage:

```
from spacerini.data import
    load_ir_dataset

hf_dset = load_ir_dataset("msmarco-
    passage")
```

We include wrappers to load database tables, pandas DataFrames (pandas development team, 2020; Wes McKinney, 2010), and text datasets on disk, as well as the ability to load any dataset either in memory-mapped mode or in streaming mode: the former makes it possible to handle larger-than-memory datasets, and the latter larger-than-disk datasets that can be streamed from a remote location such as the Hugging Face Hub.

#### 3.2 Pre-processing

Spacerini also provides a preprocess subpackage which offers a range of customizable pre-processing options for preparing datasets. This module includes a sharding utility that enables the partitioning of large datasets into smaller, more manageable chunks for efficient parallel processing.

```
from spacerini.preprocesss import
    shard_dataset

shard_dataset(
    hf_dataset=hf_dset,
    shard_size="1GB",
    column_to_index="text",
    shards_paths="msmarco-shards",
)
```

#### 3.3 Indexing

Spacerini's index subpackage leverages Pyserini to provide very efficient Lucene indexing and allow users to easily and quickly index large datasets, either sharded in the pre-processing step, or any text format accepted by Pyserini, and streaming text datasets, such as those returned by Spacerini's data subpackage. This subpackage also exposes several tokenization options using existing language-specific analyzers<sup>6</sup> as well as Hugging Face subword tokenizers (MOI et al., 2022).

```
from spacerini.index import
    index_json_shards

index_json_shards(
    shards_path="msmarco-shards",
    index_path="app/index",
)
```

<sup>6</sup>[https://lucene.apache.org/core/9\\_5\\_0/analysis/common/](https://lucene.apache.org/core/9_5_0/analysis/common/)

We also provide wrappers to Pyserini’s dense and hybrid retrieval functionality through the `spacerini.index.encode` subpackage.

### 3.4 Template-based Search Interfaces

Having indexed a collection, one can easily spin up a frontend using the frontend subpackage and one of many provided templates<sup>7</sup>. These are built using cookiecutter,<sup>8</sup> a Python templating library for software projects. We provide a few batteries-included frontend templates based both on the Gradio<sup>9</sup> and Streamlit<sup>10</sup> demo app frameworks, both of which are natively supported by Hugging Face Spaces. Figure 1 showcases a search engine built using one of our Streamlit templates.

```
from spacerini.frontend import
    create_app

cookiecutter_vars = {
    "dset_text_field": "text",
    "metadata_field": "docid",
    "space_title": "MS MARCO Search",
    "local_app": "app"
}

create_app(
    template="gradio-vanilla",
    extra_context_dict=cookiecutter_vars,
    output_dir=".",
)
```

### 3.5 Deployment to Hugging Face Spaces

The local apps developed in the previous subsection can then be pushed to Hugging Face Spaces and hosted there for free. One can then further customize the running app from the browser, for example to add functionality not provided by the chosen template. The goal of the templates is to provide a useful starting point in the form of a running app that users can further customize with interface features useful for their own workflows.

```
from spacerini.frontend import
    create_space_from_local

create_space_from_local(
    space_slug="msmarco-passage-search",
    organization="spacerini",
    space_sdk="gradio",
    local_dir=LOCAL_APP,
    delete_after_push=True,
)
```

<sup>7</sup><https://github.com/castorini/hf-spacerini/tree/main/templates>

<sup>8</sup><https://github.com/cookiecutter/cookiecutter>

<sup>9</sup><https://gradio.app/>

<sup>10</sup><https://streamlit.io/>

### 3.6 Sharing Indexes as Hugging Face Datasets

Orthogonal to the workflow presented so far, is the ability to upload Lucene indexes to the Hugging Face Hub using shareable dataset repositories and enabling reproducible retrieval experiments.

```
from spacerini.index import
    push_index_to_hub

push_index_to_hub(
    dataset_slug="lucene-english-
    analyzer-msmarco",
    index_path="index",
)
```

Any hosted index can then just as easily be downloaded for local use:

```
from spacerini.index import
    load_index_from_hub

index_path = load_index_from_hub("lucene-
fr-analyzer-")
```

### 3.7 Search and Pagination

Search features are provided by the search subpackage and leverage the memory-mapping feature of Arrow tables to load the entire table of results—no matter how big—only materializing the specific shard that corresponds to the requested result page.

```
from spacerini.search import
    result_indices, result_page

ix = result_indices(
    "Lorem Ipsum",
    num_results=1_000,
    INDEX_PATH,
)

last_results_page = result_page(
    hf_dset,
    ix,
    page=-1,
    results_per_page=20,
).to_pandas()
```

## 4 Use Cases and Demonstrations

We envision Spacerini to be useful primarily to NLP researchers, students, shared task organizers, data scientists, and data annotators, as well as tech-adjacent and -proficient professionals and laypeople. In what follows, we overview a series of 13 use-cases that we implemented and how they might benefit their respective targeted audience. An overview of all 13 search engines can be found

at <https://huggingface.co/spacerini>; in following inline links to the selected engines, this part of their URL prefix is shortened to ‘hf’ for brevity.

#### 4.1 NLP researchers

Spacerini is designed to enable qualitative analysis of large-scale textual corpora without the need for extensive engineering work. It can be used in dataset auditing campaigns, such as those carried out by [Kreutzer et al. \(2022\)](#) or in data annotation efforts. Also relevant here are generative text models whose outputs can be better understood by better understanding the datasets they were trained on. We refer the reader to Section 5 of [Piktus et al. \(2023\)](#) for a detailed exposé of potential use scenarios which include: PII detection, problematic content detection, social representation, benchmark and language contamination detection, as well as plagiarism and memorization detection. An example demo for this context is the index of the XSUM ([Narayan et al., 2018](#)) dataset which is indexed and can be explored with the [hf/xsum-search](#) demo.

#### 4.2 IR researchers

Given its tight integration with Pyserini, Spacerini can also be leveraged by IR researchers to experiment with modifications of their retrieval pipelines in user studies or to deploy demos of their working prototypes. Reproducibility for IR experiments is further enhanced thanks to the index sharing abilities showcased in Section 3.6. As a practical example, Spacerini was used in the context of the BigCode project ([Li et al., 2023](#)) to quickly experiment with multiple n-gram tokenization schemes for BM25-based code retrieval; this corresponds to the [hf/spacerini/code-search](#) demo.

#### 4.3 Linguists

Corpus linguistics relies on qualitative analyses of text corpora to understand language and its many varieties by studying the way it is used ([McEnery and Hardie, 2011](#); [Piktus et al., 2023](#)). Some of these empirical analyses can be facilitated by the usage of an inverted index, coupled with the correct querying patterns and frontend elements, both of which are easy to achieve using Spacerini.

#### 4.4 Digital Humanists

Spacerini can also be leveraged by Digital and Computational Humanists, Archivists and Librarians looking to index their collections. Indeed,

GLAM (Galleries, libraries, archives, and museums) collections are increasingly being made available as datasets. Furthermore, there is a growing interest in the digital humanities in training and using languages models, as demonstrated by the success of projects such as the *AI for Humanists Project*.<sup>11</sup> In this context, indexing data relevant to these efforts is a difficult task; often project-based and contingent upon precarious funding arrangements. Having a project-agnostic tool like Spacerini could prove valuable to this community and a useful addition to toolkits such as the GLAM Workbench ([Sherratt, 2021](#)).

#### 4.5 IR Students

Given its low barrier of entry, Spacerini can be a good tool for IR courses, where students could be tasked with developing search engines, by providing an easy-to-deploy frontend interface for their developed retrieval systems which does not even have to be deployed within the same application, as demonstrated by the [hf/chat-noir](#), a frontend wrapper for ChatNoir ([Bevendorff et al., 2018](#)).

#### 4.6 Shared task organizers

Spacerini can also be leveraged by organizers of shared tasks such as MIRACL ([Zhang et al., 2022](#)) and Touché ([Bondarenko et al., 2022](#)), who want to help participants explore the datasets without forcing them to download large volumes of data or giving participants full access to the data: it is indeed possible to host the index privately on the Hugging Face Hub and only expose access to it through a search interface. Spacerini can also be used as a platform for participants to deploy working prototypes of their submissions with a unified interface provided by the organizer as a cookiecutter template. Example demos for this use case include [hf/miracl-bengali](#), [hf/miracl-arabic](#), and [hf/miracl-swahili](#).

#### 4.7 Tech journalists

Spacerini can help data journalists and digital investigative journalists index, explore, and understand open data, in a similar vein to the functionality provided by the Aleph suite.<sup>12</sup> Providing technical tools to data journalists is a crucial in uncovering matters of public interest, as was evident by role played by the collaborative use of the Neo4j graph

<sup>11</sup><https://www.bertforhumanists.org/>

<sup>12</sup><https://docs.alephdata.org/>

database in unraveling corrupt networks surrounding tax havens (Díaz-Struck and Cabra, 2018).

#### 4.8 Additional usage patterns

Finally, three features of Hugging Face Spaces make them especially attractive for users: (1) they can leverage private datasets, meaning that one can provide search access to a dataset without sharing the underlying data, (2) they can be seamlessly embedded into HTML, specifically Gradio-based Spaces which can be embedded as *Web Components*<sup>13</sup> so that users can easily integrate a Spacerini-based search feature into their own websites<sup>14</sup>, and (3) Gradio-based Spaces expose a FastAPI<sup>15</sup> endpoint that can be queried to access the functionality of the space, making deployed search engines accessible through HTTP calls.

#### 5 Limitations and Future Plans

The main limitation of the off-the-shelf variant of Spacerini is the disk space limit imposed by Hugging Face Spaces, which is currently set to 50 GB for the free tier.<sup>16</sup> While not enough to accommodate entire corpora such as ROOTS or The Pile, such datasets are typically amalgamations of constituent datasets which can each be studied independently. This limit has no bearing on Spacerini search apps deployed locally. Should users still want to get more disk space for their Spaces-hosted indexes, they can either pay for an upgrade to a more appropriate tier or see whether they qualify for a free hardware upgrade through the community grants offered by Hugging Face, in the Settings pane of the relevant space.

Planned improvements include automating the creation of dataset cards (or rather “index cards”) when pushing an index to the Hugging Face Hub, better documentation, as well as more fine-grained tokenization support.

Please also note that Spacerini is currently in active development and that the stability of its current API and subpackages isn’t guaranteed not to involve breaking changes as we converge toward the first stable version release. We also look forward to community contributions both to the codebase and to the frontend templates, as well as in the form of

<sup>13</sup>[https://developer.mozilla.org/en-US/docs/Web/Web\\_Components](https://developer.mozilla.org/en-US/docs/Web/Web_Components)

<sup>14</sup>e.g.: <https://cakiki.github.io/search-engine/>

<sup>15</sup><https://github.com/tiangolo/fastapi>

<sup>16</sup><https://huggingface.co/docs/hub/spaces-overview#hardware-resources>

actual use cases of the library that culminate in the deployment of search apps.

#### 6 Conclusion

We presented Spacerini, a modular framework that enables the quick and free deployment and serving of template-based search indexes as interactive applications for ad-hoc exploration of text datasets. The need for such a tool is especially pressing as large language models have come to consume inordinate amounts of text data, reinforcing the need for a qualitative exploration and understanding of datasets to assess them in a way that is impenetrable to quantitative analyses alone.

Spacerini leverages features from both the Pyserini toolkit and the Hugging Face ecosystem to facilitate the creation and hosting of user-friendly search systems for text datasets. Users can easily index their collections and deploy them as ad-hoc search interfaces, making the retrieval of relevant data points a quick and efficient process. The user-friendly interface enables non-technical users to effectively search massive datasets, making Spacerini a valuable tool for anyone looking to audit their text collections qualitatively. The framework is open-source and available on GitHub under [gh/castorini/hf-spacerini](https://github.com/castorini/hf-spacerini) and demo search apps can be found under [hf/spacerini](https://hf-spacerini.com)

The key advantage of Spacerini is its ability to simplify the search process, allowing researchers to conduct quick and efficient audits, while abstracting away all the minutiae of indexing data or hosting services. We believe that this provides an opportunity for collaboration and transparency in IR and NLP research. With the creation and sharing of search indexes publicly, practitioners, researchers and the general public can work together to pinpoint problematic content, find duplicates, and identify biases in datasets.

Finally, we emphasize that Spacerini is a first step in the direction of systematic dataset auditing, and more work is still needed to create standardized structures that leverage tools such as ours to properly document the different axes of interest.

#### Acknowledgments

We are grateful to Lukas Gienapp, Daniel van Strien, Yuvraj Sharma, Omar Sanseviero, Julien Chaumond, Lucain Pouget, Abubakar Abid, and Leandro von Werra for their tireless support and invaluable advice throughout this project.

## 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*.
- 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. *CoRR*, abs/2212.04960.
- Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Anne Beaulieu and Sabina Leonelli. 2021. *Data and Society: A Critical Introduction*. Sage.
- 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, FAccT '21*, page 610–623, New York, NY, USA. Association for Computing Machinery.
- Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York. Springer.
- Alexander Bondarenko, Maik Fröbe, Johannes Kiesel, Shahbaz Syed, Timon Gurcke, Meriem Beloucif, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2022. Overview of touché 2022: Argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5-8, 2022, Proceedings*, volume 13390 of *Lecture Notes in Computer Science*, pages 311–336. Springer.
- Emilia Díaz-Struck and Mar Cabra. 2018. *Uncovering International Stories with Data and Collaboration*, pages 55–65. Springer International Publishing, Cham.
- Ethan Fast, Tina Vachovsky, and Michael S. Bernstein. 2016. Shirtless and dangerous: Quantifying linguistic signals of gender bias in an online fiction writing community. *CoRR*, abs/1603.08832.
- Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. *CoRR*, abs/1802.00393.
- 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.
- Tony Hey, Stewart Tansley, Kristin Tolle, and Jim Gray. 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research.
- 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. *CoRR*, abs/2203.15556.
- Ben Hutchinson, Vinodkumar Prabhakaran, Emily Denton, Kellie Webster, Yu Zhong, and Stephen Denuyl. 2020. Social biases in NLP models as barriers for persons with disabilities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5501, Online. Association for Computational Linguistics.
- Eun Seo Jo and Timnit Gebru. 2020. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT\* '20*, page 306–316, New York, NY, USA. Association for Computing Machinery.
- 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, Iroro 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.
- Ulrich Krohs. 2012. Convenience experimentation. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, 43(1):52–57. Data-Driven Research in the Biological and Biomedical Sciences On Nature and Normativity: Normativity, Teleology, and Mechanism in Biological Explanation.

- 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*.
- Sabina Leonelli. 2020. Scientific Research and Big Data. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Summer 2020 edition. Metaphysics Research Lab, Stanford University.
- 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, 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 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. 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 Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi 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!](#) *CoRR*, abs/2305.06161.
- 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.
- Tony McEnery and Andrew Hardie. 2011. *Corpus Linguistics: Method, Theory and Practice*. Cambridge Textbooks in Linguistics. Cambridge 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.
- Anthony MOI, Nicolas Patry, Pierric Cistac, Pete, Funtowicz Morgan, Sebastian Pütz, Mishig, Bjarte Johansen, Thomas Wolf, Sylvain Gugger, Clement, Julien Chaumond, Lysandre Debut, François Garillot, Luc Georges, dctelus, JC Louis, MarcusGrass, Taufiqzaman Peyash, Oxflotus, Alan deLevie, Alexander Mamaev, Arthur, Cameron, Colin Clement, Dagmawi Moges, David Hewitt, Denis Zolotukhin, and Geoffrey Thomas. 2022. [huggingface/tokenizers: Rust 0.13.2](#).
- Jakob Mökander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. 2023. [Auditing large language models: a three-layered approach](#).
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. [MS MARCO: A human generated machine reading comprehension dataset](#). In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- 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, Benoît Sagot, and Laurent Romary. 2019. [Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures](#). In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom. Leibniz-Institut für Deutsche Sprache.
- The pandas development team. 2020. [pandas-dev/pandas: Pandas](#).
- Aleksandra Piktus, Christopher Akiki, Paulo Villegas, Hugo Laurençon, Gérard Dupont, Sasha Luccioni, Yacine Jernite, and Anna Rogers. 2023. [The ROOTS search tool: Data transparency for LLMs](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 304–314, Toronto, Canada. 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. *J. Mach. Learn. Res.*, 21(1).
- Tim Sherratt. 2021. [Glam workbench](#).
- Shoaib Ahmed Siddiqui, Nitarshan Rajkumar, Tegan Maharaj, David Krueger, and Sara Hooker. 2022. [Metadata archaeology: Unearthing data subsets by leveraging training dynamics](#). *CoRR*, abs/2209.10015.
- 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.
- Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. [Ethical and social risks of harm from language models](#). *CoRR*, abs/2112.04359.
- Wes McKinney. 2010. [Data Structures for Statistical Computing in Python](#). In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.
- 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.
- Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamaloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2022. [Making a miracl: Multilingual information retrieval across a continuum of languages](#).



# Adapters: A Unified Library for Parameter-Efficient and Modular Transfer Learning

Clifton Poth<sup>\*1,4</sup>, Hannah Sterz<sup>\*1</sup>, Indraneil Paul<sup>1</sup>,  
Sukannya Purkayastha<sup>1</sup>, Leon Engländer<sup>1</sup>, Timo Imhof<sup>1</sup>,  
Ivan Vulić<sup>2</sup>, Sebastian Ruder<sup>3</sup>, Iryna Gurevych<sup>1</sup>, Jonas Pfeiffer<sup>3</sup>

<sup>1</sup>Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt

<sup>2</sup>Language Technology Lab, University of Cambridge

<sup>3</sup>Google DeepMind <sup>4</sup>Cohere

## Abstract

We introduce *Adapters*, an open-source library that unifies parameter-efficient and modular transfer learning in large language models. By integrating 10 diverse adapter methods into a unified interface, *Adapters* offers ease of use and flexible configuration. Our library allows researchers and practitioners to leverage adapter modularity through composition blocks, enabling the design of complex adapter setups. We demonstrate the library’s efficacy by evaluating its performance against full fine-tuning on various NLP tasks. *Adapters* provides a powerful tool for addressing the challenges of conventional fine-tuning paradigms and promoting more efficient and modular transfer learning. The library is available via <https://adapterhub.ml/adapters>.

## 1 Introduction

The ever-increasing size of pretrained large language models (LLMs) (Brown et al., 2020; Chowdhery et al., 2022) has made the established transfer learning paradigm of fine-tuning all model parameters on a downstream task (Howard and Ruder, 2018; Devlin et al., 2019) extremely expensive. Moreover, the requirement of *parameter efficiency* at fine-tuning, while definitely paramount, is not the only shortcoming of the predominant LLM fine-tuning paradigm. It also suffers from other crucial issues such as negative interference, lack of positive transfer between tasks in multi-task learning (McCloskey and Cohen, 1989), catastrophic forgetting (French, 1999), and poor generalization.

Two closely related lines of research aimed at addressing this set of challenges have gained significant attention recently. First, *parameter-efficient fine-tuning* (Lialin et al., 2023; Sabry and Belz, 2023) focuses on the aspect of computational efficiency and feasibility by only fine-tuning a small

number of parameters for downstream tasks. Second, *modular transfer learning* (Pfeiffer et al., 2023) focuses on the aspect of knowledge transfer by designing self-contained network modules which can be aggregated for better multi-task performance and generalization. In practice, these often represent two sides of the same coin. Methods that devise small components within a language model for fine-tuning on labeled task data, henceforth generally denoted as *adapters*, are both parameter-efficient *and* modular in nature.

The initial release of *AdapterHub* (Pfeiffer et al., 2020a) marks the first attempt to systematically make adapters accessible to researchers and practitioners in an easy-to-use framework. AdapterHub proposed a framework to easily integrate, train and use adapters for state-of-the-art Transformer models with minimal changes. It additionally established an open platform to share, explore and consume pre-trained adapter modules. While AdapterHub focused on bottleneck-style adapters (Houlsby et al., 2019) initially, the field of adapter methods has expanded substantially since (Li and Liang, 2021; Mahabadi et al., 2021; Hu et al., 2022; He et al., 2022; Liu et al., 2022, among others).

With increasing interest in adapter methods, new tools and libraries have been developed. *OpenDelta* (Hu et al., 2023a), HuggingFace’s *PEFT* (Mantrik et al., 2022) and *LLM-Adapters* (Hu et al., 2023b) are recent examples of libraries which attempt to unify adapter methods in a single code base and extend their applicability to new model architectures. However, these works exclusively focus on the parameter-efficiency aspect of adapters, neglecting the modularity side of these methods.

**Contributions.** Based on the initial version of AdapterHub, we, therefore, propose *Adapters*, a new library aimed at *unifying parameter-efficient and modular transfer learning*. Compared to the first AdapterHub iteration and concurrent libraries, our main contributions can be summarized as fol-

\*Authors contributed equally.

lows: **1)** We propose a self-contained library that integrates 10 diverse adapter methods into a unified interface for easy usage and flexible configuration; **2)** we develop a simple way of leveraging the modularity of adapters by designing *composition blocks* that allow flexibly defining complex adapter setups; **3)** we integrate all methods into 20 Transformer-based models spanning NLP, vision, and multi-modal applications; **4)** we evaluate the performance of our adapter implementations against full fine-tuning on a diverse set of tasks.

## 2 Background

We use the term *adapter* in a more general sense to refer to a broad family of transfer learning methods that share the two defining properties: parameter efficiency and modularity. For a detailed overview of different adapter architectures, we refer the reader to the recent survey by Pfeiffer et al. (2023).

### 2.1 Parameter Efficiency

Let the parameters of a language model be composed of a set of pre-trained parameters  $\Theta$  (frozen) and a set of parameters  $\Phi$  (where  $\Phi$  can either be newly introduced or  $\Phi \subset \Theta$ ). During fine-tuning, adapter methods optimize only  $\Phi$  according to a loss function  $L$  on a dataset  $D$ :

$$\Phi^* \leftarrow \arg \min_{\Phi} L(D; \{\Theta, \Phi\})$$

Different adapter methods insert parameters  $\Phi$  at different locations of a pre-trained large model. Bottleneck adapters (Rebuffi et al., 2017; Houlsby et al., 2019), as one of the early methods, introduce bottleneck feed-forward layers in each layer of a Transformer model. Subsequent designs have adapted a Transformer model’s self-attentions (Li and Liang, 2021), bias terms (Ben Zaken et al., 2022), input prompts (Lester et al., 2021) or embeddings (Pfeiffer et al., 2021b). Complementary lines of work have focused on optimizing the parameter efficiency (Mahabadi et al., 2021; Liu et al., 2022) and runtime efficiency (Hu et al., 2022; Lei et al., 2023) of adapters or have attempted to unify multiple components in a single framework (He et al., 2022; Mao et al., 2022).

### 2.2 Modularity

A modular deep learning model is composed of modules that each capture a specific functionality of the full model, such as task or language capacities. Pfeiffer et al. (2023) propose a taxonomy

|                                    | AdapterHub v1        | Adapters                      |
|------------------------------------|----------------------|-------------------------------|
| Design                             | Fork of Transformers | Self-contained add-on library |
| Adapter methods                    | 2                    | 10                            |
| Complex configurations             | ✗                    | ✓                             |
| Composition blocks                 | ✗ <sup>1</sup>       | ✓ (6)                         |
| Model architectures                | 3                    | 20                            |
| AdapterHub.ml / HF Hub integration | ✓ / ✗                | ✓ / ✓                         |

Table 1: Feature comparison between the initial *AdapterHub* release (Pfeiffer et al., 2020a) and the proposed *Adapters* library.

of modular deep learning methods covering the dimensions of computation function, routing, aggregation, and training.

Routing and aggregation are of special interest here as they coordinate the composition of multiple adapter modules, a key functionality enabled by modularity. Exemplary existing work includes using stochastic routing through adapters (Wang et al., 2022), adapter parameter averaging (Friedman et al., 2021), sequential function aggregation of adapter modules (Pfeiffer et al., 2022b) as well as weighted (Wang et al., 2021) and attention-based (Pfeiffer et al., 2021a) output aggregation.

Finally, along the training dimension, the modularity of adapters allows using pre-trained adapter modules as initialization for further fine-tuning (Poth et al., 2021; Vu et al., 2022).

## 3 The Adapters Library

*Adapters* builds on many design decisions established in the initial *AdapterHub* release (Pfeiffer et al., 2020a), but offers substantial extensions both ‘horizontally’ (e.g., extending the support to many more pretrained neural architectures, extending the coverage of adapter architectures) and ‘vertically’ (e.g., adding new composition and processing capabilities). Table 1 gives an overview of the differences between the initial *AdapterHub* and *Adapters*. The core features adopted from the initial release, facilitating its ease of use and wider adoption by researchers and practitioners, include: **1)** Tight integration into the widely used HuggingFace Transformers (Wolf et al., 2020) library; **2)** adaptation of pre-existing Transformers fine-tuning scripts with minimal changes; **3)** single-line saving and loading of adapter modules from a shared community hub.

<sup>1</sup>V1 already supported stacking and fusing adapter, however without flexibly composable blocks.

```

import adapters
from transformers import AutoModel

model = AutoModel.from_pretrained("..")
adapters.init(model)
model.add_adapter("a", config="seq_bn")
model.add_adapter("b", config="seq_bn")
model.train_adapter(Parallel("a", "b"))

```

Listing 1: Example of adding adapters to an existing Transformers model. After model instantiation, `init()` introduces adapter-specific functionality. Two bottleneck adapters are added via `add_adapter()` and activated for parallel training.

### 3.1 Transformers Integration

Unlike the initial AdapterHub, *Adapters* is designed as a standalone package that acts as an add-on to the Transformers library. *Adapters* provides adapter implementations and management methods that can be injected into pre-trained Transformer checkpoints without modifying the original model code directly. We provide two approaches to this end: (i) by attaching to existing models and (ii) by providing our own, specialized model classes.

**Attaching to Models.** The `init()` method provides a straightforward solution for making adapters accessible to pre-existing model classes post-hoc. A model checkpoint for one of the supported architectures (cf. § 3.5) can be instantiated via any of the model classes provided by the Transformers library. An example of this approach is given in Listing 1. All adapter-related functionality is injected post-instantiation by passing the model instance to the `init()` method. Afterwards, all methods provided by *Adapters* are easily invocable from the same model instance.

**AdapterModel Classes.** As an alternative to post-hoc initialization, *Adapters* provides a set of built-in model classes optimized for working with adapters. Following HuggingFace’s naming conventions, these classes are named `XXXAdapterModel`, where `XXX` is the name of the model architecture. Compared to models with attached adapters, these classes especially provide more flexibility with regard to prediction heads. Each model instance can have multiple named prediction heads targeted towards different tasks loaded simultaneously. These prediction heads can be associated with adapter modules by sharing a common name.

Providing this functionality is crucial for enabling features such as quickly switching between

| Method name                                 | Default config  |
|---|-----------------|
| Bottleneck adapter (Houlsby et al., 2019)   | [double_]seq_bn |
| Invertible adapter (Pfeiffer et al., 2020b) | seq_bn_inv      |
| Prompt tuning (Lester et al., 2021)         | prompt_tuning   |
| Prefix tuning (Li and Liang, 2021)          | prefix_tuning   |
| Compacter (Mahabadi et al., 2021)           | compacter       |
| LoRA (Hu et al., 2022)                      | lora            |
| (IA) <sup>3</sup> (Liu et al., 2022)        | ia3             |
| Parallel adapter (He et al., 2022)          | par_bn          |
| Mix-and-Match adapter (He et al., 2022)     | mam             |
| UniPELT (Mao et al., 2022)                  | unipelt         |

Table 2: Overview of adapter methods supported in the *Adapters* library at the time of submission (Aug 2023).

adapters targeted toward different tasks at runtime. It is also essential for creating composed adapter setups such as parallel inference on multiple tasks (cf. § 3.4). We, therefore, provide automatic conversion from HuggingFace’s model classes - typically paired with a single, fixed prediction head - to our newly introduced classes featuring flexible prediction heads.

### 3.2 Unified Adapter Interface

*Adapters* defines a common interface of methods covering the full life cycle of working with adapters. This includes methods for adding, activating, saving, releasing, loading, aggregating, and deleting adapter modules. When adding a new adapter to a model (e.g., via `add_adapter()`), it is given a unique identifier string. All adapter-related methods then solely use this string to identify the adapter module an operation should be performed on. Thus, the adapter interface at the model level can be agnostic to specific adapter methods.

This interface, as well as adapter implementations at the module level, are integrated into model classes via Python mixins and dynamically modifying Python classes at runtime to keep changes to the existing Transformers code base minimal.

### 3.3 Adapter Methods

Each adapter method is defined by a configuration object or string which allow flexible customization of various properties of an adapter module, including placement, capacity, residual connections, initialization etc. We distinguish between single methods consisting of one type of adapter module and complex methods consisting of multiple different adapter module types. Table 2 gives an overview of all methods currently integrated into *Adapters*, along with their configuration strings.

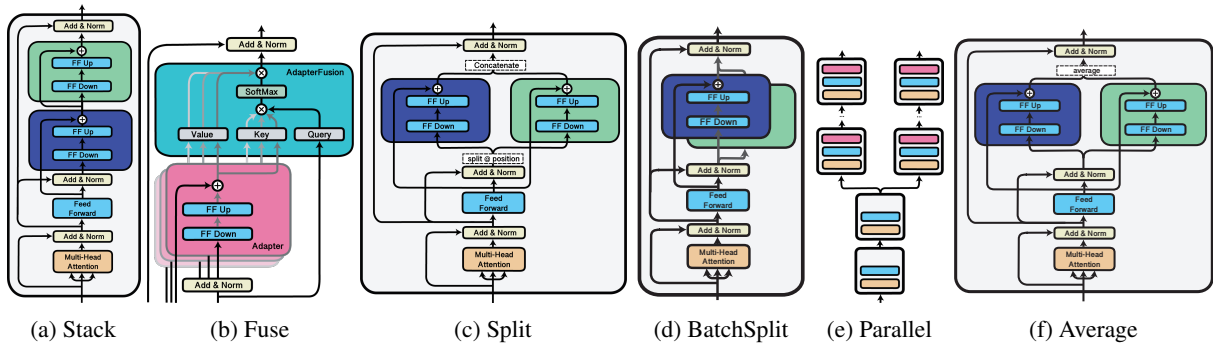


Figure 1: Overview of adapter composition blocks supported in the *Adapters* library at the time of writing this paper.

### 3.3.1 Single Methods

*Adapters* supports single adapter methods that introduce parameters in new feed-forward modules such as bottleneck adapters (Houlsby et al., 2019), introduce prompts at different locations such as prefix tuning (Li and Liang, 2021), reparameterize existing modules such as LoRA (Hu et al., 2022) or re-scale their output representations such as (IA)<sup>3</sup> (Liu et al., 2022). Detailed descriptions for all currently implemented single methods are given in Appendix A, see also Table 2 again.

### 3.3.2 Complex Methods

While different efficient fine-tuning methods and configurations have often been proposed as standalone, combining them for joint training has proven to be beneficial (He et al., 2022; Mao et al., 2022). To make this process easier, *Adapters* provides the possibility to group multiple configuration instances using the `ConfigUnion` class. This flexible mechanism allows easy integration of multiple complex methods proposed in the literature (as the two examples outlined below), as well as the construction of other, new complex configurations currently not available nor benchmarked in the literature (Zhou et al., 2023).

**Mix-and-Match Adapters (He et al., 2022)** was proposed as a combination of Prefix-Tuning and parallel bottleneck adapters. Using `ConfigUnion`, this method can be defined as:

```
config = ConfigUnion(
    PrefixTuningConfig(bottleneck_size=800),
    ParallelConfig(),
)
model.add_adapter("name", config=config)
```

**UniPELT (Mao et al., 2022)** combines LoRA, Prefix Tuning, and bottleneck adapters in a single unified setup. It additionally introduces a gating mechanism that controls the activation of the different adapter modules.  $\mathcal{G}_m \leftarrow \sigma(W_{G_m} \cdot x)$ .

### 3.4 Adapter Composition

While the modularity and composability aspects of adapters have seen increasing interest in research, existing open-source libraries (Mangrulkar et al., 2022; Hu et al., 2023a) have largely overlooked these aspects. *Adapters* makes adapter compositions a central and accessible part of working with adapters by enabling the definition of complex, composed adapter setups. We define a set of simple *composition blocks* that each capture a specific method of aggregating the functionality of multiple adapters. Each composition block class takes a sequence of adapter identifiers plus optional configuration as arguments. The defined adapter setup is then parsed at runtime by *Adapters* to allow for dynamic switching between adapters per forward pass. Fig. 1 shows schematic illustrations of all composition blocks supported by *Adapters*. Listing 2 shows examples of how the same composition blocks are defined in code:

```
Stack("a", "b", "c")
Fuse("d", "e", "f")
Split("g", "h", splits=[64, 64])
BatchSplit("i", "j", batch_sizes=[2, 4])
Parallel("k", "l", "m")
Average("n", "o", weights=[0.3, 0.7])
Stack("p", Parallel("q", "r"))
```

Listing 2: Code examples of composition blocks supported by *Adapters*. Strings represent adapter IDs.

In what follows, we present each supported composition in more detail.

**Stack.** The Stack block allows stacking multiple adapters sequentially within a Transformer layer. This type of composition is, e.g., used in the MAD-X framework for cross-lingual transfer (Pfeiffer et al., 2020b), where language and task adapters are stacked. In Listing 2, the input is first passed through *a*, the output of *a* is then inputted to *b*, and the output of *b* is finally inputted to *c*.

**Fuse.** The Fuse block can be used to activate an AdapterFusion layer (Pfeiffer et al., 2021a). AdapterFusion is a non-destructive way to combine the knowledge of multiple pre-trained adapters on a new downstream task. In Listing 2, we activate the adapters  $d$ ,  $e$ , and  $f$  as well as the fusion layer that combines the outputs of all three.<sup>2</sup>

**Split.** The Split block can be used to split an input sequence between multiple adapters. This e.g. enables splitting multimodal input sequences to modality-specific adapters (Pfeiffer et al., 2022a). In Listing 2, we split each input sequence between adapters  $g$  and  $h$ . All tokens with indices 0 - 63 are forwarded through  $g$  while the next 64 tokens beginning at index 64 are forwarded through  $h$ .

**BatchSplit.** The BatchSplit block splits inputs along the batch size dimension between several adapters. That is, different adapters receive different sub-batches of the full input batch. In Listing 2, we split the input batch between adapters  $i$ , and  $j$ . Adapter  $i$  receives two sequences and  $j$  receives four sequences. The sum of all specified sub-batches has to match the batch size of the input.

**Parallel.** This block can be used to enable independent parallel training and inference on different adapters, where each adapter has its own prediction head. The implementation automatically replicates all inputs at the first occurrence of parallel adapter modules, sharing the inputs in all lower layers without parallel modules. This mechanism was first used in Rücklé et al. (2021). In Listing 2, we forward all inputs via adapters  $k$ ,  $l$ , and  $m$  in parallel.

**Average.** Following approaches of ensembling full models at inference time for better generalization, recent work has explored methods of averaging pre-trained adapters. This includes averaging adapter output representations (Wang et al., 2021) as well as averaging adapter parameters (Friedman et al., 2021; Wang et al., 2022; Chronopoulou et al., 2023). *Adapters* provides built-in support for both types of inference time-averaging methods. **Output averaging** allows to dynamically aggregate the output representations of multiple adapters in a model forward pass via weighted averaging. This is realized via the Average composition block. In Listing 2, two adapters are averaged with the weights 0.3 for  $n$  and 0.7 for  $o$ . **Parameter averaging** enables creating a new adapter via

<sup>2</sup>Note that this requires a fusion layer to be added beforehand via `add_adapter_fusion()`.

weighted averaging of the parameters of multiple pre-trained adapters. As this process is typically not done dynamically at runtime, *Adapters* provides `average_adapter()` as a dedicated method. Compared to output averaging, parameter averaging of adapters has the advantage of not inducing any additional inference time relative to using a single adapter.

**Nesting.** Finally, it is possible to nest composition blocks within other composition blocks to create deeper and more complex compositions. *Adapters* defines a set of allowed nestings to restrict the users to setups that are sensible. As an example, we nest a Parallel block within a Stack block in Listing 2.

### 3.5 Supported Models

At the time of release, *Adapters* has built-in support for 20 widely adopted model architectures included in the Transformers library. This covers text encoder models such as BERT (Devlin et al., 2019) and DeBERTa (He et al., 2021), text decoder models such as GPT-2 (Radford et al., 2019), sequence-to-sequence models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), vision encoder models such as ViT (Dosovitskiy et al., 2021), as well as multimodal models such as CLIP (Radford et al., 2021).<sup>3</sup>

While adapter-related implementations mostly can be shared across all supported models, correctly integrating them into each model implementation requires manual effort. While it is difficult to standardize this process due to differences between model architectures, we provide clear guidelines for integrating new models in the form of shared interfaces and step-by-step documentation<sup>4</sup>.

### 3.6 AdapterHub Ecosystem

*Adapters* is integrated into the extensive existing open-source ecosystem introduced by AdapterHub (Pfeiffer et al., 2020a). Most prominently, this includes `AdapterHub.ml` as a platform to share and discover pre-trained adapter modules. *Adapters* further broadens the possibilities for sharing adapters by integrating with HuggingFace’s Model Hub, which has emerged as one of the primary platforms for open-sourcing model checkpoints. The new Hub integration comes with programmatic methods of discovering and publishing pre-trained adapter

<sup>3</sup>An up-to-date full list of supported models can be found at [https://docs.adapterhub.ml/model\\_overview.html](https://docs.adapterhub.ml/model_overview.html).

<sup>4</sup>[https://docs.adapterhub.ml/contributing/adding\\_adapters\\_to\\_a\\_model.html](https://docs.adapterhub.ml/contributing/adding_adapters_to_a_model.html)

| Method                  | Sequence Classification  |                   |                         |                          |                   |                          |                         | Regression        | Mult. Choice             | Extract. QA             | Tagging                  | Avg.                     |
|-------------------------|--------------------------|-------------------|-------------------------|--------------------------|-------------------|--------------------------|-------------------------|-------------------|--------------------------|-------------------------|--------------------------|--------------------------|
|                         | CoLA<br>Dev MCC          | MNLI<br>Dev Acc.  | MRPC<br>Dev F1          | QNLI<br>Dev Acc.         | QQP<br>Dev F1     | RTE<br>Dev Acc.          | SST-2<br>Dev Acc.       | STS-B<br>Dev PCC  | Cosmos QA<br>Dev Acc.    | SQuAD v2<br>Dev F1      | CoNLL-2003<br>Test F1    |                          |
| double_seq_bn           | 63.58<br>(±19.19)        | 87.61<br>(±26.41) | <b>93.31</b><br>(±4.52) | 92.84<br>(±17.17)        | 91.58<br>(±36.83) | <b>80.87</b><br>(±11.09) | 94.73<br>(±17.51)       | 90.85<br>(±27.16) | 70.99<br>(±16.87)        | <b>84.89</b><br>(±5.52) | 91.92<br>(±17.65)        | <b>85.74</b><br>(±18.17) |
| seq_bn                  | <b>71.22</b><br>(±23.40) | 87.5<br>(±20.39)  | 92.91<br>(±4.54)        | <b>93.15</b><br>(±15.83) | 89.69<br>(±21.31) | 79.42<br>(±9.81)         | 95.18<br>(±13.26)       | 89.44<br>(±20.33) | 69.68<br>(±16.44)        | 82.88<br>(±1.04)        | <b>92.02</b><br>(±11.48) | <b>85.74</b><br>(±14.34) |
| par_bn                  | 63.95<br>(±23.72)        | 87.44<br>(±21.66) | 93.24<br>(±4.65)        | 93.04<br>(±17.26)        | 88.32<br>(±33.14) | 77.98<br>(±10.95)        | 94.95<br>(±16.97)       | 90.33<br>(±5.64)  | <b>80.10</b><br>(±18.47) | 82.56<br>(±6.70)        | 91.95<br>(±27.60)        | 85.81<br>(±16.98)        |
| compacter               | 55.52<br>(±13.67)        | 86.10<br>(±1.99)  | 90.43<br>(±3.58)        | 92.42<br>(±2.68)         | 86.68<br>(±2.14)  | 68.59<br>(±4.91)         | 94.15<br>(±0.81)        | 90.06<br>(±23.27) | 67.91<br>(±10.42)        | 79.20<br>(±8.87)        | 91.27<br>(±8.58)         | 82.03<br>(±7.36)         |
| prefix_tuning           | 61.62<br>(±4.93)         | 86.98<br>(±18.91) | 91.06<br>(±4.09)        | 92.46<br>(±9.55)         | 87.07<br>(±15.58) | 71.12<br>(±6.06)         | <b>95.18</b><br>(±0.54) | 90.13<br>(±29.23) | 66.13<br>(±3.44)         | 78.16<br>(±2.41)        | 91.46<br>(±2.44)         | 82.85<br>(±8.79)         |
| lora                    | 63.99<br>(±20.64)        | 87.59<br>(±4.29)  | 92.60<br>(±4.39)        | 93.11<br>(±3.77)         | 88.48<br>(±2.57)  | 80.26<br>(±9.28)         | 94.99<br>(±8.48)        | 90.72<br>(±19.31) | 70.63<br>(±8.65)         | 82.46<br>(±8.86)        | 91.85<br>(±21.68)        | 85.15<br>(±10.17)        |
| ia3                     | 63.03<br>(±21.39)        | 86.19<br>(±5.08)  | 92.32<br>(±3.94)        | 91.88<br>(±3.73)         | 86.41<br>(±13.46) | 76.89<br>(±7.17)         | 94.42<br>(±2.13)        | 90.65<br>(±29.16) | 66.85<br>(±9.69)         | 78.52<br>(±10.11)       | 91.56<br>(±21.94)        | 83.52<br>(±11.62)        |
| <b>Full Fine-tuning</b> | 63.66                    | <b>87.63</b>      | 90.20                   | 92.81                    | <b>91.92</b>      | 78.77                    | 94.81                   | <b>91.20</b>      | 68.87                    | 82.91                   | 91.33                    | 84.91                    |

Table 3: Best performance ( $\pm$  std. dev. across all hyper-parameters) of various supported single adapter methods (cf. §3.3.1) applied to roberta-base, benchmarked against full fine-tuning. The best results and lowest std. dev. per task are highlighted in **bold**.

modules, in addition to the previously available methods for downloading and saving adapters.

At the time of writing, users of *Adapters* have access to over 700 pre-trained adapters.

## 4 Adapter Evaluation

In addition to the ease of use aforementioned, we show that the adapter methods offered by our library are performant across a range of settings. To this end, we conduct evaluations on the single adapter implementations made available by *Adapters* (see §3.3.1). We demonstrate the effectiveness of these methods against full fine-tuning on a variety of task types: extractive question answering (Rajpurkar et al., 2018), multiple choice classification (Huang et al., 2019), sequence tagging (Tjong Kim Sang, 2002), sequence to sequence summarization (Narayan et al., 2018), sequence classification and sequence regression (Wang et al., 2019). To make our evaluations reflective of user experience, we conduct them using the two most commonly used base model variants on AdapterHub: *roberta-base* (Liu et al., 2019) and *bart-base* (Lewis et al., 2020).

**Setup.** We conduct a grid search over a range of common training hyper-parameters, varying learning rates between  $\{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-4}, 10^{-3}\}$  and the number of epochs between  $\{5, 10, 20, 30\}$ . We also augment the grid with a number of adapter-specific hyper-parameters. These, along with the minimum and maximum trainable parameters added across the configurations, are detailed in Table A. The highest attained performance (and the standard deviation of results across the grid) for the two chosen base models are outlined in Tables 3 and 4, respectively.

**Results.** The obvious takeaway from our evaluations is that all adapter implementations offered by our framework are competitive with full model fine-tuning, across all task classes. Approaches that offer more tunable hyper-parameters (and thus allow for easy scaling) such as Bottleneck adapters, LoRA, and Prefix Tuning predictably have the highest topline performance, often surpassing full fine-tuning. However, extremely parameter-frugal methods like (IA)<sup>3</sup>, which add  $< 0.005\%$  of the parameters of the base model, also perform commendably and only fall short by a small fraction. Finally, the Compacter is the least volatile among the single methods, obtaining the lowest standard deviation between runs on the majority of tasks.

## 5 Conclusion

We have presented *Adapters*, a novel library for research and application of adapters. Unlike comparable solutions, *Adapters* equally focuses on the parameter-efficiency and modularity side of adapters. Our library implements a diverse set of adapter methods under a unified interface which allows flexible configuration and mixing of different approaches. We proposed a simple building block system for leveraging the modularity of adapters to build complex adapter setups. *Adapters* tightly integrates into the HuggingFace and AdapterHub ecosystems and its adapter implementations show performances competitive to full fine-tuning.

As research on adapters and LLMs continues to advance rapidly, our library will evolve as well. Its extensibility makes *Adapters* well prepared for the integration of new adapter methods and model architectures, both from us and the community.

## Acknowledgements

This work has been funded by Huawei Technologies (Ireland) Co., Ltd. and German Research Foundation (DFG) as part of the Research Training Group KRITIS No. GRK 2222. We thank Martin Hentschel for his help building the demo web app for our library. We thank Francesco Piccinno and Srini Narayanan for their helpful comments and suggestions during the initial drafts of this paper. We also thank the many contributors and users of the *AdapterHub* open-source framework.

## References

- Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. [BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland.
- 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*.
- 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 Pillai, 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](#). *ArXiv preprint*.
- Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. 2023. [AdapterSoup: Weight averaging to improve generalization of pre-trained language models](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2054–2063, Dubrovnik, Croatia.
- 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.
- 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.
- Robert M. French. 1999. [Catastrophic forgetting in connectionist networks](#). *Trends in Cognitive Sciences*, (4):128–135.
- 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.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. [Towards a unified view of parameter-efficient transfer learning](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [Deberta: decoding-enhanced bert with disentangled attention](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.
- 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*, Proceedings of Machine Learning Research, pages 2790–2799.
- Jeremy Howard and Sebastian Ruder. 2018. [Universal language model fine-tuning for text classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia.

- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.
- Shengding Hu, Ning Ding, Weilin Zhao, Xingtai Lv, Zhen Zhang, Zhiyuan Liu, and Maosong Sun. 2023a. [OpenDelta: A plug-and-play library for parameter-efficient adaptation of pre-trained models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 274–281, Toronto, Canada.
- Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023b. [Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models](#). *ArXiv preprint*.
- Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. [Cosmos QA: Machine reading comprehension with contextual commonsense reasoning](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China.
- Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Y. Zhao, Yuexin Wu, Bo Li, Yu Zhang, and Ming-Wei Chang. 2023. [Conditional adapters: Parameter-efficient transfer learning with fast inference](#). *ArXiv preprint*.
- 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.
- 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.
- 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.
- Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. [Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning](#). *ArXiv preprint*.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohata, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. [Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, Nov 28 - Dec 9, 2022, virtual and New Orleans, LA, USA*, pages 1950–1965.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Ro{bert}a: A robustly optimized {bert} pretraining approach](#). *ArXiv preprint*.
- Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. [Compacter: Efficient low-rank hypercomplex adapter layers](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1022–1035.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. [Peft: State-of-the-art parameter-efficient fine-tuning methods](#). <https://github.com/huggingface/peft>.
- Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. [UniPELT: A unified framework for parameter-efficient language model tuning](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland.
- Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.
- Jonas Pfeiffer, Gregor Geigle, Aishwarya Kamath, Jan-Martin Steitz, Stefan Roth, Ivan Vulić, and Iryna Gurevych. 2022a. [xGQA: Cross-lingual visual question answering](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2497–2511, Dublin, Ireland.
- Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022b. [Lifting the curse of multilinguality by pre-training modular transformers](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States.



- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. [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.
- 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.
- Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo M. Ponti. 2023. [Modular Deep Learning](#). *ArXiv preprint*.
- 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.
- Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. [UNKs everywhere: Adapting multilingual language models to new scripts](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic.
- Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. [What to pre-train on? Efficient intermediate task selection](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10585–10605, Online and Punta Cana, Dominican Republic.
- 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 *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 8748–8763.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#). *Technical report*.
- 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.*, pages 140:1–140:67.
- 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.
- Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. [Learning multiple visual domains with residual adapters](#). 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 506–516.
- 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 the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic.
- Mohammed Sabry and Anya Belz. 2023. [Peft-ref: A modular reference architecture and typology for parameter-efficient finetuning techniques](#). *ArXiv preprint*.
- Erik F. Tjong Kim Sang. 2002. [Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition](#). In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.
- Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou’, and Daniel Cer. 2022. [SPoT: Better frozen model adaptation through soft prompt transfer](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.
- Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021. [Efficient test time adapter ensembling for low-resource language varieties](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 730–737, Punta Cana, Dominican Republic.
- Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. [AdaMix: Mixture-of-adaptations for parameter-efficient model tuning](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, Abu Dhabi, United Arab Emirates.
- 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. [Trans-formers: 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.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. 2023. [AutoPEFT: Automatic configuration search for parameter-efficient fine-tuning](#). *ArXiv preprint*.

## A Description of Adapter Methods

**Bottleneck Adapters** introduce bottleneck modules in each layer of a Transformer model. Generally, these adapter modules consist of a down-projection matrix  $W_{down}$  that projects into a lower dimension  $d_{bottleneck}$ , a non-linearity  $f$ , an up-projection  $W_{up}$  that projects back into the original hidden layer dimension and a residual connection  $r$ , i.e.:  $h \leftarrow W_{up} \cdot f(W_{down} \cdot h) + r$ .

Depending on the specific configuration, these layers can be introduced at different locations and in **sequential** or **parallel** order relative to the adapted Transformer layer. *Adapters* provides pre-defined configurations for the sequential configurations of Houshy et al. (2019) and Pfeiffer et al. (2021a) as well as the parallel configuration of He et al. (2022).

**Invertible Adapters** were proposed as part of MAD-X (Pfeiffer et al., 2020b) to learn language-specific transformations. Embedding outputs are passed through an invertible adapter module in the forward direction before entering the first Transformer layer and in the inverse direction after leaving the last Transformer layer.

**Prompt Tuning (Lester et al., 2021)** is an approach to condition language models on a task-specific soft prompt. While hard prompts consist of fixed textual descriptions prepended to the model’s input (Brown et al., 2020), these soft prompts are continuously optimized towards the target task via gradient descent. Prompt tokens are prepended to the embedded input sequence.

**Prefix Tuning (Li and Liang, 2021)** is an extension of prompt tuning which prepends trainable prefix vectors  $P^K$  and  $P^V$  to the keys and values of the multi-head attention block inputs. The prefix vectors have a configurable length and are not optimized directly but reparameterized via a bottleneck MLP in the built-in default configuration, following the original implementation.

**Compacter (Mahabadi et al., 2021)** exchanges the linear down- and up-projection of a bottleneck adapter for a PHM layer<sup>5</sup>. This PHM layer constructs its weight matrix from two smaller matrices, which reduces the number of parameters needed for the adapters. These matrices can be factorized and shared between all adapter layers. *Adapters* provides pre-defined configurations for the Compacter and Compacter++ variants.

**LoRA (Hu et al., 2022)** injects trainable low-rank decomposition matrices into the layers of a pre-trained model. For any model layer expressed as a matrix multiplication of the form  $h = W_0x$ , it performs a reparameterization such that:  $h = W_0x + \frac{\alpha}{r}BAx$ . Here,  $A \in \mathbb{R}^{r \times k}$  and  $B \in \mathbb{R}^{d \times r}$  are the decomposition matrices and  $r$  is the low-dimensional rank of the decomposition. With *Adapters*, LoRA modules can be configured to be placed into the self-attention, intermediate, or output components of a Transformer layer. Following Hu et al. (2022), *Adapters* provides a built-in method of merging LoRA modules with the original pre-trained weights of a model for inference without additional latency.

**(IA)<sup>3</sup> (Liu et al., 2022)** introduces trainable vectors  $l_W$  into different components of a Transformer model, which perform element-wise rescaling of inner model activations. For any model layer expressed as a matrix multiplication of the form  $h = Wx$ , it therefore performs an element-wise multiplication with  $l_W$ , such that:  $h = l_W \odot Wx$ .

<sup>5</sup>Parametrized hypercomplex multiplication layer.

| Method                  | Sequence Classification |                         |                   |                  |                   | Regression        | Extract. QA       | Seq2Seq                 | Avg.              |
|-------------------------|-------------------------|-------------------------|-------------------|------------------|-------------------|-------------------|-------------------|-------------------------|-------------------|
|                         | CoLA                    | MRPC                    | QNLI              | RTE              | SST-2             | STS-B             | SQuAD v2          | XSum                    |                   |
|                         | Dev MCC                 | Dev F1                  | Dev Acc.          | Dev Acc.         | Dev Acc.          | Dev PCC           | Dev F1            | Rouge 2                 |                   |
| double_seq_bn           | 61.80<br>(±21.47)       | 91.62<br>(±3.73)        | 92.18<br>(±15.35) | 73.65<br>(±8.26) | 94.04<br>(±13.89) | 89.89<br>(±23.18) | 79.37<br>(±12.17) | 17.49<br>(±1.51)        | 75.00<br>(±10.86) |
| seq_bn                  | 53.29<br>(±18.82)       | 91.16<br>(±3.59)        | 92.11<br>(±6.29)  | 75.45<br>(±6.84) | 93.46<br>(±1.09)  | 89.44<br>(±20.38) | 79.47<br>(±10.41) | 17.76<br>(±1.60)        | 74.02<br>(±7.55)  |
| par_bn                  | 55.58<br>(±19.70)       | 90.75<br>(±3.46)        | 92.12<br>(±14.42) | 75.09<br>(±7.91) | 94.08<br>(±14.39) | 89.58<br>(±10.86) | 79.63<br>(±8.19)  | <b>18.26</b><br>(±1.19) | 74.39<br>(±10.01) |
| compacter               | 44.81<br>(±12.06)       | 87.44<br>(±4.30)        | 91.45<br>(±3.14)  | 69.68<br>(±3.85) | 93.35<br>(±1.18)  | 87.62<br>(±14.09) | 74.33<br>(±9.81)  | 13.55<br>(±5.29)        | 70.28<br>(±6.72)  |
| prefix_tuning           | 48.57<br>(±8.23)        | <b>92.31</b><br>(±3.65) | 91.61<br>(±6.97)  | 75.95<br>(±5.63) | 93.58<br>(±1.92)  | 90.27<br>(±13.72) | 80.45<br>(±8.73)  | 16.81<br>(±4.94)        | 73.69<br>(±6.73)  |
| lora                    | 55.94<br>(±21.93)       | 91.61<br>(±4.17)        | 92.53<br>(±4.31)  | 76.05<br>(±8.67) | 94.03<br>(±1.31)  | 89.57<br>(±33.03) | 78.81<br>(±10.47) | 17.21<br>(±1.13)        | 74.47<br>(±10.63) |
| ia3                     | 50.65<br>(±13.64)       | 89.68<br>(±2.79)        | 90.80<br>(±7.29)  | 72.92<br>(±5.83) | 94.22<br>(±6.04)  | 89.05<br>(±21.27) | 72.46<br>(±8.46)  | 13.51<br>(±4.08)        | 71.66<br>(±8.68)  |
| <b>Full Fine-tuning</b> | <b>62.80</b>            | 90.4                    | <b>94.9</b>       | <b>87.0</b>      | <b>96.6</b>       | <b>91.2</b>       | <b>89.2</b>       | 17.73                   | <b>86.84</b>      |

Table 4: Best performance ( $\pm$  std. dev. across all hyper-parameters) of various supported single adapter methods (refer §3.3.1) applied to facebook/bart-base, benchmarked against full fine-tuning. The best results and lowest std. dev. per task are highlighted in **bold**.

| PEFT Method   | Attribute Name   | Range               | Added Params. |                   |
|---------------|------------------|---------------------|---------------|-------------------|
|               |                  |                     | Min           | Max               |
| double_seq_bn | reduction_factor | {2, 16, 64}         | 461,088       | <b>14,183,424</b> |
| seq_bn        | reduction_factor | {2, 16, 64}         | 230,544       | 7,091,712         |
| par_bn        | reduction_factor | {2, 16, 64}         | 230,544       | 7,091,712         |
| compacter     | reduction_factor | {4, 16}             | 58,816        | 69,184            |
|               | phm_dim          | {4, 8}              |               |                   |
| prefix_tuning | bottleneck_size  | {32, 128, 512}      | 636,704       | 10,002,944        |
|               | prefix_length    | {5, 50, 200}        |               |                   |
| lora          | r                | {4, 8, 16, 64, 200} | 147,456       | 7,372,800         |
| ia3           | —                | —                   | <b>55,296</b> | 55,296            |

Table 5: Adapter-specific attributes (name as in framework) and their values that we used for grid-search along with the minimum and maximum possible parameters added over the resultant grid. Extreme values are highlighted in **bold**.

## B Adapter Evaluation Results

Best task performance for all task–adapter method combinations are presented in Table 3 with RoBERTa as base and model and in Table 4 with BART as base model. All adapter methods provided by *Adapters* are competitive to full fine-tuning. Table 5 gives an overview of the evaluated adapter configurations. Fig. 2 plots best performing learning rate–capacity combinations for evaluated single adapter methods. Lower capacity adapters perform better with higher learning rates.

Fig. 3 plots mean task performance by learning rate for evaluated single adapter methods. For bottleneck adapters, the best-performing learning rate is  $10^{-4}$ . Compacter performs best with a learning rate of  $10^{-3}$ , prefix tuning with  $3 \cdot 10^{-4}$ , LoRA with  $5 \cdot 10^{-4}$ , and (IA)<sup>3</sup> with  $5 \cdot 10^{-3}$ . These results align with the learning rates proposed by the respective adapter method authors.

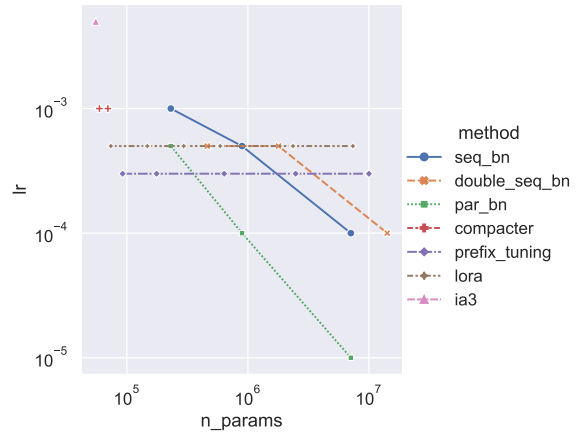


Figure 2: Best performing learning rate–capacity combinations for evaluated single adapter methods. Lower capacity adapters perform better with higher learning rates.

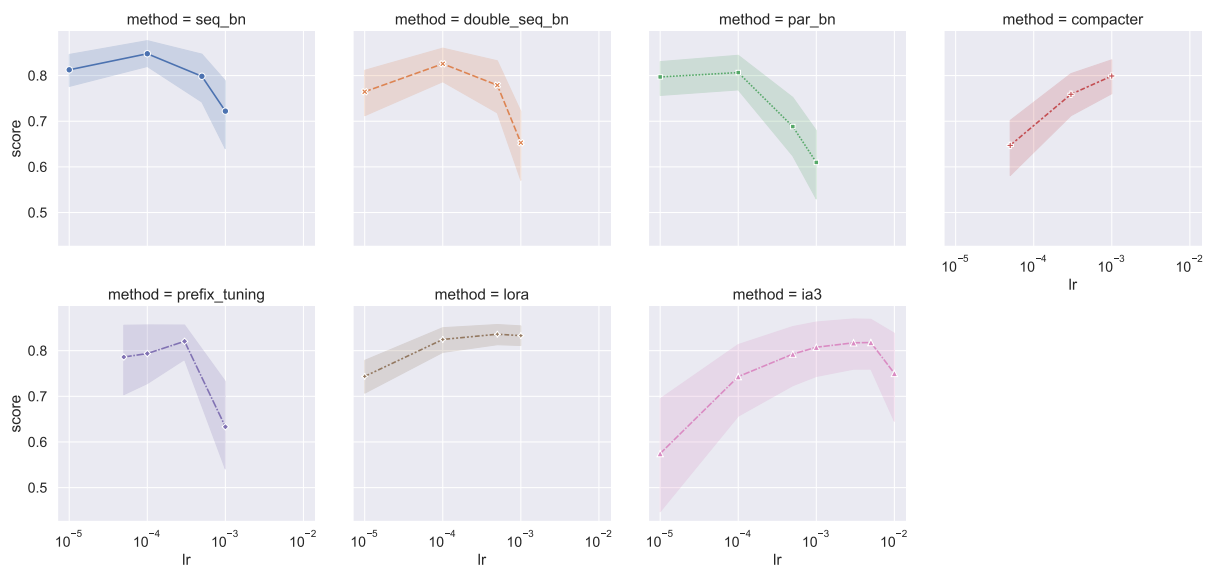


Figure 3: Mean task performance by learning rate for evaluated single adapter methods.

# INTELMO: Enhancing Models' Adoption of Interactive Interfaces

Chunxu Yang<sup>†</sup>, Chien-Sheng Wu<sup>§</sup>, Lidiya Murakhovs'ka<sup>§</sup>,  
Philippe Laban<sup>§</sup>, Xiang 'Anthony' Chen<sup>†</sup>

<sup>†</sup> UCLA HCI Research, <sup>§</sup> Salesforce Research

{chunxuyang, xac}@ucla.edu

{wu.jason, l.murakhovska, plaban}@salesforce.com

## Abstract

This paper presents INTELMO, an easy-to-use toolkit to help model developers adopt user-facing interactive interfaces for their language models. The toolkit provides default style patterns over interaction-based categorization, ensuring that developers can build fully interactive interfaces with minimal and intuitive additional code. Moreover, INTELMO employs a multi-granular hierarchical abstraction to provide developers with flexible control over the generation process. INTELMO is under active development, with document available at <https://intelmo.github.io/>

## 1 Introduction

As natural language processing (NLP) and human-computer interaction (HCI) theories advance, the demand for integrating the two has markedly increased. However, a significant challenge persists in bridging the gap between NLP model development and user interaction. This predicament arises due to the divergent skill sets of these two groups: while model developers excel in building advanced algorithms, they may lack expertise in developing complex interactive user interfaces. Conversely, front-end developers might possess proficiency in crafting engaging user interfaces, but they may lack the knowledge of the intricacies involved in NLP model development (Cai and Guo, 2019).

This situation leads to a critical need for model developers to rapidly implement interactive user interfaces. The conventional software development workflow, where model developers pass their projects to front-end engineers to design the user interface, may not be feasible at the model-tuning stage.

In addition, efforts have been made to ensure that NLP models use datasets that closely align with real-world tasks. Chandu et al. (2021) highlights the significance of bridging the gap between common NLP datasets and real-world sce-

narios, proposing a method to enhance authenticity through dynamic grounding. Additionally, several models, such as TWEETNLP (Camacho-collados et al., 2022), MARVISTA (Chen et al., 2023) and RESTGPT (Song et al., 2023), utilized real web data as their training and testing data source.

However, building a web scraper from scratch poses a daunting task for model developers. During the model's tuning and optimization process, developers require real-time feedback on the model's performance and may also need to compare or combine outputs from multiple models. A seamless flow of real-world data can significantly enhance the model's interpretability and enable developers to preview how their model will perform when delivered to end-users.

Considering the current situation, we believe that a toolkit generating interactive interfaces for model developers should have the following characteristics, listed in order of importance:

1. **Interactivity:** The toolkit should allow interaction with the model's interface and provide real-time feedback. It should also include a configurable module to adjust model parameters, enabling the observation of the model's performance under different settings.
2. **Flexibility:** The toolkit should support most common NLP tasks and facilitate interaction with multiple models. Additionally, it should provide fine-grained control over the display of model results.
3. **Usability:** The toolkit should abstract application programming interfaces (APIs) for model developers, ensuring that Python developers without web knowledge can integrate the system into their models with minimal code.
4. **Automation:** The toolkit should automatically crawl information from the real-time

web and stream it into the model with no efforts from developers.

This demo introduces INTELMO, a versatile toolkit that employs multiple layers of abstraction to meet all requirements listed above. (1) INTELMO is built on Flask<sup>1</sup>, serving as a foundation to offer HTML templates and a flexible building environment. Through encapsulating default styles, this toolkit can effortlessly construct interfaces based on model developer configurations. (2) The toolkit divides articles into three nested levels: *paragraphs*, *sentences*, and *words*. Each level comes with specific APIs for customization and default styles tailored to different task types. (3) Simplifying the developer’s task, INTELMO adopts task categorization. By specifying the task type such as *Modification*, *Generation*, or *Insertion* based on interaction process (detailed classification in Section 3.2.1), developers can initiate a complete web application using within ten lines of non-intrusive code. (4) INTELMO automatically retrieves the Really Simple Syndication (RSS) source from configurable news websites in the background once the application is launched, freeing developers from the burden of constructing scrapers.

We believe that tools like INTELMO can significantly reduce the difficulty of building and deploying interactive interfaces for NLP models, thereby improving model performance, customization, and interpretability through iterative feedback.

## 2 Related Work

In recent years, as various NLP models have gained extensive attention, researchers have started to take notice of evaluation metrics beyond language model performance. Wang et al. (2021) emphasized the importance of Human-in-the-loop (HITL) NLP framework, while ITG (Faltings et al., 2023), and WEBGPT (Nakano et al., 2022) incorporated human interaction into *text generation*, and *question answering (QA)* tasks within the NLP domain. Gu et al. (2023), Lee et al. (2023) and Carta et al. (2023) have also recognized the significance of models being applicable to real-world data, proposing evaluation criteria based on real-world human-language model interactions. DMS (Jónsson and Loftsson, 2023) and WEBSHOP (Yao et al., 2023) attempted to integrate different NLP

<sup>1</sup><https://flask.palletsprojects.com/>

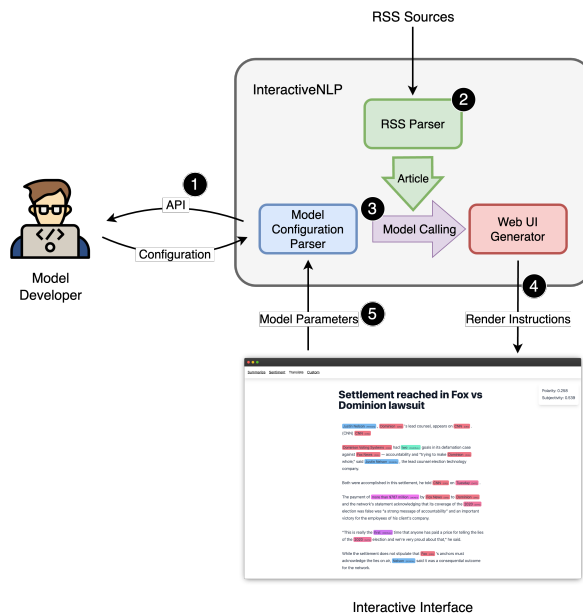


Figure 1: The basic structure of INTELMO: (1) The toolkit provides encapsulated API interfaces to model developers. (2) INTELMO utilizes an embedded RSS parser to fetch RSS information and parse it into articles. (3) The system maps model configurations to corresponding functions and passes the articles to the model function. (4) INTELMO renders the interactive interface based on the model’s results. (5) End-users or model developers can modify the parameters through forms on the webpage. The modified model parameters are incorporated into the model function. (Icon made by @surang from [www.flaticon.com](http://www.flaticon.com).)

tasks to achieve dynamic delivery of multi-task systems.

Regarding relevant tools and platforms, EXPLAINBOARD (Liu et al., 2021) and ADAPTER-HUB (Beck et al., 2022) serves as no-code platforms for developing and testing language models. More recently, IFAN (Mosca et al., 2023) utilizes API technology to enable real-time, interpretation-based interactions with models.

## 3 INTELMO

We created **IN**terface **T**oolkit for **EX**ensible **L**anguage **M**odels (**INTELMO**) as a solution for generating interactive interfaces for NLP models. As shown in Figure 1, the system consists of the following components:

1. **Model Configuration Parser:** This component is responsible for providing control interfaces to model developers. The specific configuration abstractions will be detailed in Section 3.2.

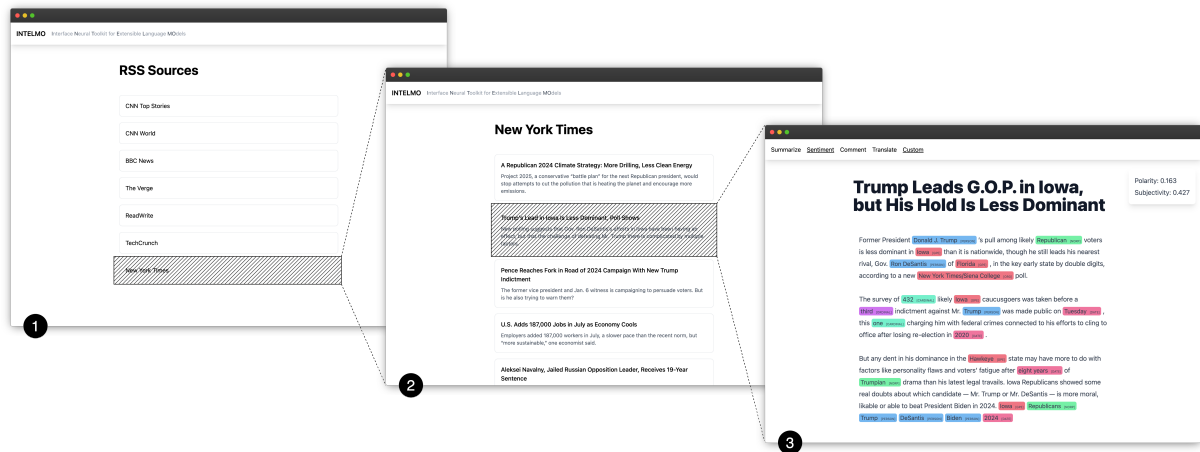


Figure 2: The user interface of INTELMO. (1) RSS sources page; (2) Article list page; (3) NLP reader page.

2. **RSS Parser:** This module implements the retrieval of real-world web data. The RSS Parser automatically collects articles from different sources based on configuration settings. After preprocessing, these articles are passed as parameters into the model.
3. **Web UI Generator:** This module generates rendering instructions based on the results returned by the model, creating interactive UI components. The precise control over page elements will be elaborated on in Section 3.1.

These components are embedded within INTELMO. Unless a model developer decides to finely control each step, which the API allows and provides for, they do not need to understand the specific operational details. For a basic workflow, a model developer only needs to pass configuration options to INTELMO which requires usually no more than 10 lines of code per feature, and the generated interactive pages will be produced.

### 3.1 User Interface

The user interface of INTELMO, as shown in Figure 2, resembles an RSS reader. The interface relies on Flask with Jinja template engine<sup>2</sup> and is styled using TailwindCSS<sup>3</sup>. The initial UI generated by INTELMO serves as the RSS source page. On this page, users can select sources of interest from pre-configured RSS feeds. Upon selection, INTELMO employs a parser to extract the list of articles. The list page displays article titles and brief descriptions.

<sup>2</sup><https://jinja.palletsprojects.com/>

<sup>3</sup><https://tailwindcss.com/>

The reader page of INTELMO is illustrated in Figure 3. After entering this page, users can apply the model functions from the top-left corner. Hovering the cursor over the function name enables the adjustment of available model parameters.

As a core feature for model developers, INTELMO provides relevant configuration options for most page elements. For each element, model developers can set its content or configure child elements at the next level. In the latter case, INTELMO prioritizes rendering the child elements. If model developers are not satisfied with the provided built-in styles, they can specify Tailwind-CSS class names or even directly include HTML tags within the element’s content for complete customization.

### 3.2 Configuration

INTELMO offers model developers a multi-layered abstraction to define tasks and build UI systems. Through this approach, the toolkit provides developers with various default styles and task types, while also allowing customization at each level.

#### 3.2.1 Categorization of NLP Tasks Based on Interaction

NLP tasks are often divided based on the internal processes of the tasks (Dudhabaware and Madankar, 2014), which provides a clear structure for constructing evaluation metrics within specific categories. However, in the context of interactive interfaces, this approach becomes overly specific and less conducive to abstraction. For example, *Sentiment Analysis* and *Question Generation* over entire articles might belong to completely differ-

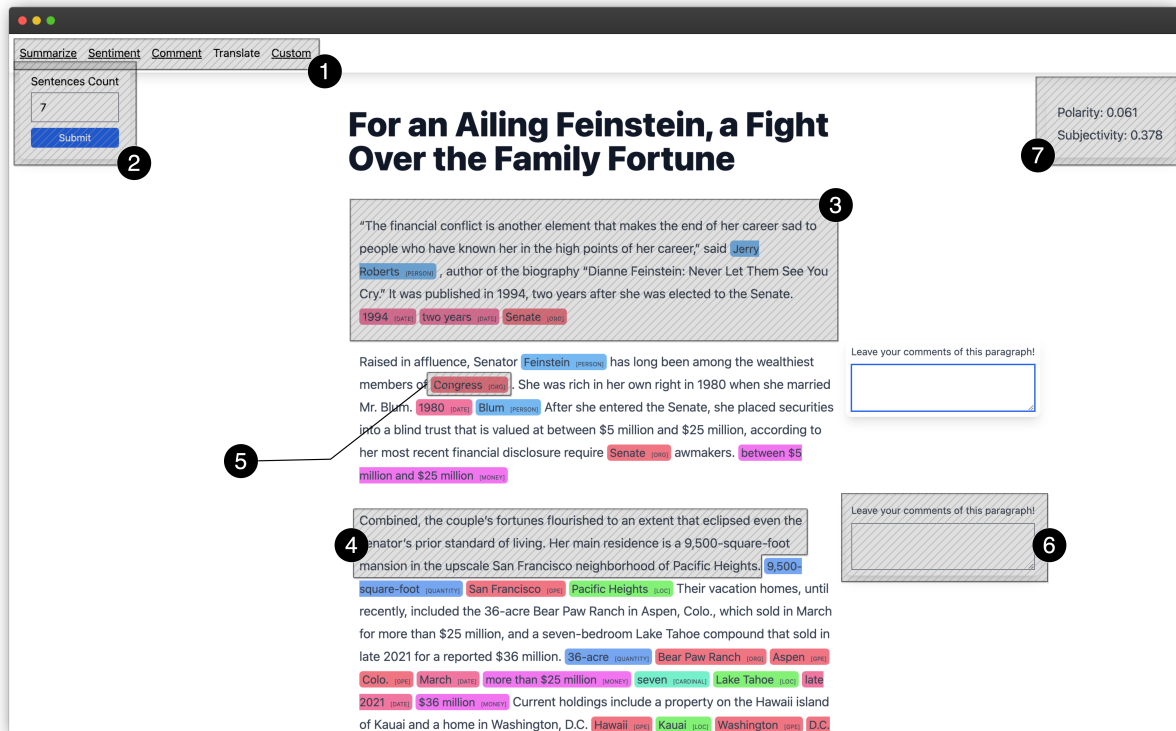


Figure 3: The UI of INTELMO has the following controllable elements: (1) Model function list; (2) Model parameters form; (3) Paragraph-level elements; (4) Sentence-level elements; (5) Word-level elements; (6) Extra elements of paragraphs; (7) Global elements over the entire article.

ent task categories under traditional classification. Nevertheless, the interactive operations required by these models are quite similar, which is generating one or more new page elements based on potential parameters and displaying them on the page. From an interactive perspective, these two tasks should be grouped together and abstracted using a unified approach.

By adopting this categorization, it's expected that model developers would generalize models from the perspective of human-model interaction. This approach can reduce the complexity of creating interactive interfaces, simplifying the workflow and APIs.

Based on the interaction behavior of models, we proposed the following categories for NLP tasks, as shown in Figure 4:

1. **Modification:** Making changes to the existing elements rather than adding new ones to the article. An example of this is *Information Filtering*, which means highlighting certain content within articles while fading other parts.
2. **Insertion:** Adding new paragraphs, sentences, or words to the article. This classification re-

sults in a change in the structure of the article's content. An example is *Machine Translation*.

3. **Generation:** Generating new information based on parts or the entirety of the article. This new information is displayed outside the article. One example of this is *Sentiment Analysis*.
4. **Cross-document Tasks:** Tasks of this nature may require access to or interaction with other articles, including *Named Entity Linking* and *Information Aggregation*. INTELMO provides specialized APIs for such tasks.

The NLP tasks depicted in Figure 4 are not exhaustive. Based on the characteristics of different categories, new tasks can be easily classified. Additionally, INTELMO offers custom task types. By specifying this type in the configuration, models can access the entire RSS article information and precisely control the rendering of results.

### 3.2.2 Task Composition

If a model developer needs to showcase multiple tasks on a single page, INTELMO offers the following types of compositions:



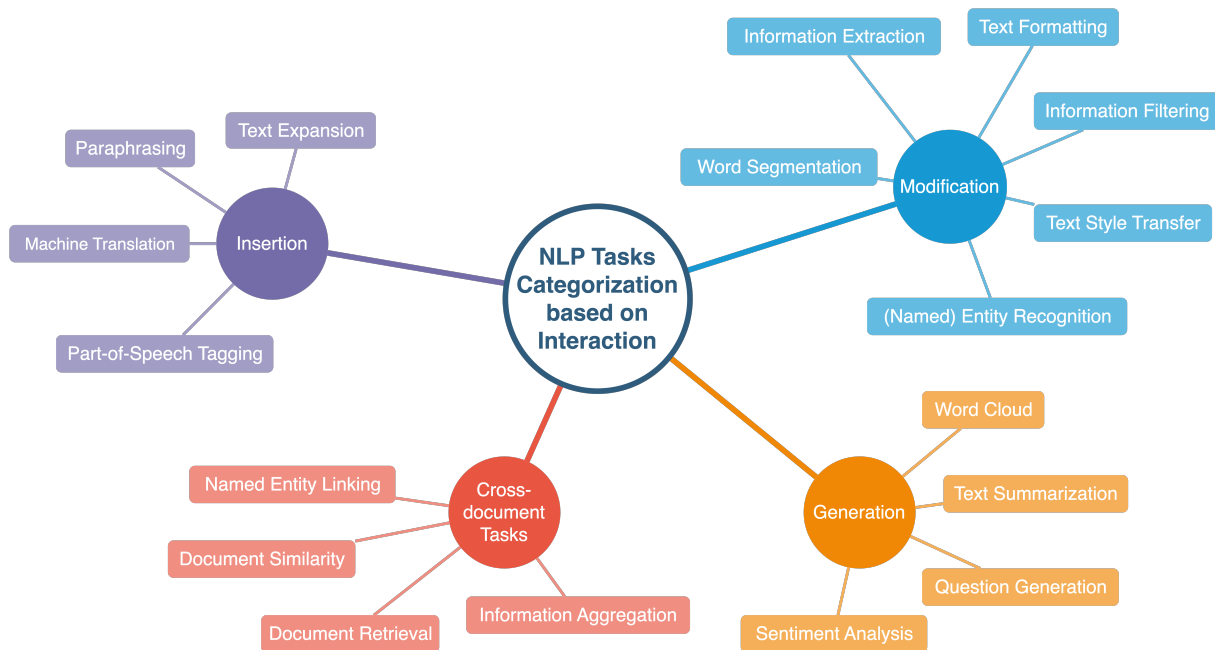


Figure 4: The classification of NLP Tasks based on interaction produces 4 basic categories: (1) Modification; (2) Insertion; (3) Generation; (4) Cross-document Tasks.

1. **Exclusive:** This mode disallows simultaneous execution of different tasks. When one task is activated, others are disabled.
2. **Compatible:** Multiple tasks can run concurrently. Each task gets the same original article. INTELMO takes care of rendering the combined results on the page.
3. **Pipelined:** Different tasks are connected sequentially, where each task function receives the output of the previous task function as its input. This mode is effective in systems with a complete workflow like MARVISTA (Chen et al., 2023).

These compositions could be applied recursively, allowing developers to achieve complex control flows by specifying nested compositions.

#### 4 Discussion & Future Work

Traditional model interaction testing and deployment often require developers to possess web development skills or collaborate with front-end engineers. This decreases the efficiency of model development and testing, limiting the applicability of many models. INTELMO aims to bridge this gap by enhancing developers’ development experience and efficiency through real-world datasets and a comprehensive UI framework. However, INTELMO is not currently ready to assist in build-

ing interactive pages for large-scale, long-latency complex models. Given the need to accommodate the requirements of all models, reducing user wait times will be a key focus for INTELMO’s future optimization efforts.

Furthermore, we also recognize that obstacles faced by model developers include deploying models to existing web services such as Vercel<sup>4</sup> and Google Cloud Platform<sup>5</sup> and setting up CI/CD pipelines. We aspire to continue iterating on INTELMO and provide streamlined model deployment solutions.

#### References

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. 2022. *AdapterHub Playground: Simple and Flexible Few-Shot Learning with Adapters*. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–75, Dublin, Ireland. Association for Computational Linguistics.

Carrie J. Cai and Philip J. Guo. 2019. *Software developers learning machine learning: Motivations, hurdles, and desires*. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 25–34.

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

<sup>5</sup><https://cloud.google.com/>

- Jose Camacho-collados, Kiamehr Rezaee, Talayeh Riahi, Asahi Ushio, Daniel Loureiro, Dimosthenis Antypas, Joanne Boisson, Luis Espinosa Anke, Fangyu Liu, and Eugenio Martínez Cámara. 2022. [TweetNLP: Cutting-edge natural language processing for social media](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–49, Abu Dhabi, UAE. Association for Computational Linguistics.
- Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. [Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning](#). ArXiv:2302.02662 [cs].
- Khyathi Raghavi Chandu, Yonatan Bisk, and Alan W Black. 2021. [Grounding ‘Grounding’ in NLP](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4283–4305, Online. Association for Computational Linguistics.
- Xiang ‘Anthony’ Chen, Chien-Sheng Wu, Lidiya Murakhovs’ka, Philippe Laban, Tong Niu, Wenhao Liu, and Caiming Xiong. 2023. [Marvista: Exploring the Design of a Human-AI Collaborative News Reading Tool](#). ArXiv:2207.08401 [cs].
- Rahul S. Dudhabaware and Mangala S. Madankar. 2014. [Review on natural language processing tasks for text documents](#). In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–5.
- Felix Faltings, Michel Galley, Baolin Peng, Kianté Brantley, Weixin Cai, Yizhe Zhang, Jianfeng Gao, and Bill Dolan. 2023. [Interactive Text Generation](#). ArXiv:2303.00908 [cs].
- Yu Gu, Xiang Deng, and Yu Su. 2023. [Don’t Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments](#). ArXiv:2212.09736 [cs].
- Haukur Páll Jónsson and Hrafn Loftsson. 2023. [DMS: A System for Delivering Dynamic Multitask NLP Tools](#). In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, pages 504–510.
- Mina Lee, Megha Srivastava, Amelia Hardy, John Thickett, Esin Durmus, Ashwin Paranjape, Ines Gerard-Ursin, Xiang Lisa Li, Faisal Ladhak, Frieda Rong, Rose E. Wang, Minae Kwon, Joon Sung Park, Hancheng Cao, Tony Lee, Rishi Bommasani, Michael Bernstein, and Percy Liang. 2023. [Evaluating Human-Language Model Interaction](#). ArXiv:2212.09746 [cs].
- Pengfei Liu, Jinlan Fu, Yang Xiao, Weizhe Yuan, Shuaicheng Chang, Junqi Dai, Yixin Liu, Zihuiwen Ye, Zi-Yi Dou, and Graham Neubig. 2021. [ExplainsBoard: An Explainable Leaderboard for NLP](#). ArXiv:2104.06387 [cs].
- Edoardo Mosca, Daryna Dementieva, Tohid Ebrahim Ajdari, Maximilian Kummeth, Kirill Gringauz, and Georg Groh. 2023. [IFAN: An Explainability-Focused Interaction Framework for Humans and NLP Models](#). ArXiv:2303.03124 [cs].
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. [WebGPT: Browser-assisted question-answering with human feedback](#). ArXiv:2112.09332 [cs].
- Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. [RestGPT: Connecting Large Language Models with Real-World Applications via RESTful APIs](#). ArXiv:2306.06624 [cs].
- Zijie J. Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. [Putting Humans in the Natural Language Processing Loop: A Survey](#). In *Proceedings of the First Workshop on Bridging Human-Computer Interaction and Natural Language Processing*, pages 47–52, Online. Association for Computational Linguistics.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023. [WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents](#). ArXiv:2207.01206 [cs].

## A Supplementary Files

The code is open-sourced under MIT license on Github<sup>6</sup>. A supplementary video demo is hosted at Vimeo<sup>7</sup>.

<sup>6</sup><https://github.com/INTELM0/intelmo/>

<sup>7</sup><https://vimeo.com/852034145/>

# Humanoid Agents: Platform for Simulating Human-like Generative Agents

**Zhilin Wang\***  
University of Washington  
and NVIDIA  
zhilinw@uw.edu

**Yu Ying Chiu\***  
University of Washington  
kellycyy@uw.edu

**Yu Cheung Chiu**  
The University of Hong Kong  
jackycyc@connect.hku.hk

## Abstract

Just as computational simulations of atoms, molecules and cells have shaped the way we study the sciences, true-to-life simulations of human-like agents can be valuable tools for studying human behavior. We propose Humanoid Agents, a system that guides Generative Agents to behave more like humans by introducing three elements of System 1 processing: Basic needs (e.g. hunger, health and energy), Emotion and Closeness in Relationships. Humanoid Agents are able to use these dynamic elements to adapt their daily activities and conversations with other agents, as supported with empirical experiments. Our system is designed to be extensible to various settings, three of which we demonstrate, as well as to other elements influencing human behavior (e.g. empathy, moral values and cultural background). Our platform also includes a Unity WebGL game interface for visualization and an interactive analytics dashboard to show agent statuses over time. Our platform is available on <https://www.humanoidagents.com/> and code is on <https://github.com/HumanoidAgents/HumanoidAgents>.

## 1 Introduction

Ushered in by the landmark paper on Generative Agents (Park et al., 2023), the promise of modelling conceivable human behavior using advanced NLP systems has sparked the imagination of many. Generative Agents plan activities over a day, execute them at each time-step and adapt their plans based on observations of their environment. While this approach can generate seemingly believable activities to external observers, this process does not fully resemble how humans think. Most of us do not create plans well in advance, and then meticulously and precisely carry out those plans in day-to-day life. Instead, we constantly adapt our

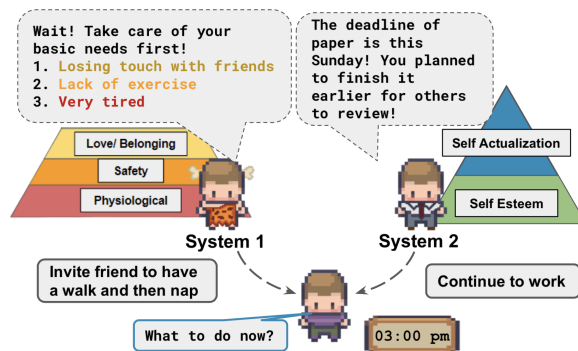


Figure 1: Humanoid Agents are guided by both System 1 thinking to respond to their embodied conditions such as their basic needs and System 2 thinking involving explicit planning.

plans to how we feel on the inside, in addition to changes in our physical environment.

To mitigate this shortcoming, we draw inspiration from psychology to propose Humanoid Agents. Kahneman (2011) suggests that humans have two complementary processes for thinking: System 1 is intuitive, effortless and instantaneous while System 2 is logical, intentional and slow. Generative Agents focus on System 2 thinking at the cost of System 1. To better guide the behavior of Humanoid Agents using System 1, we introduce three aspects of System 1 that can influence their behavior: Basic needs, Emotion and the Closeness of their social relationship with other agents.

Basic needs refer to intrinsic needs that humans have for survival (Maslow, 1943). To appropriately model human behavior, agents need to interact with others, maintain their health and rest. Failing to do so adequately, agents will receive negative feedback comprising loneliness, sickness and tiredness, as illustrated in Fig. 1. System 2 planning alone can implicitly account for activities to meet these needs (e.g. planning time for rest) but without feedback from System 1, agents cannot adapt by having a nap at 3pm if tired but bedtime is planned

\*Equal Contribution

for midnight. Similarly, a realistic model of human behavior needs to consider the agents’ emotions (Ekman, 1992). If an agent is feeling angry, it should be able to respond by doing something that helps it to vent its emotion, such as going for a run or doing meditation.

The relationship closeness of an agent to other agents should also influence how they engage with other agents. The social brain hypothesis proposes that a large part of our cognitive ability evolved to track the quality of social relationships (Dunbar, 2009), which means that people often adapt their interactions with others based on how close they feel to them (Zhou et al., 2005; Miller, 2012). To better imitate humans, we empower humanoid agents to adapt their conversations with one another based on how close they are to one another.

We present a platform that can simulate the behavior of Humanoid Agents in various settings (three of which we demonstrate), visualize them using an Unity WebGL game interface and present the statuses of simulated agents over time using an interactive analytics dashboard. We then show experiments that validate how Humanoid Agents effectively respond to and infer changes in each aspect of System 1. While our paper demonstrates how three different aspects of System 1 influence agent behavior, our system is also extensible to many more aspects, such as personality (Wu et al., 2020), moral values (Jiang et al., 2022), empathy (Sharma et al., 2020), helpfulness (Wang and Torres, 2022), cultural background (Liu et al., 2021) and other personal attributes (Wang et al., 2022).

## 2 Related Work

**Building Agents using LLMs** Humanoid Agents directly build upon Generative Agents, which aim to model believable human behavior (Park et al., 2023). To the best of our knowledge, this is the only work that seek to model day-to-day activities of human-like agents, rather than activities targeted towards achieving an externally defined goal. Liu et al. (2023) proposed simulated agents with long-term memory to align agent responses (to assistant-type prompts) with those of other agents, aiming to cooperatively improve the overall model’s ability to follow instructions. Langchain Agents (Chase, 2022), BabyAGI (Yohei, 2022), AutoGPT (Significant, 2023), AgentVerse (OpenBMB, 2023), Voyager (Wang et al., 2023) and CAMEL (Li et al., 2023) seek to create task-oriented agents that can

recursively decompose user-defined tasks into simpler sub-tasks and then solve them.

**Persona-Grounded Dialogue** Prior works have been done to ground multi-turn dialogue response generation on emotions (Rashkin et al., 2019), game character descriptions (Jack Urbanek, 2019) and personal facts (Zhang et al., 2018). However, these dialogues are not situated in a dynamic simulation of agents (which also perform activities at each time-step), and instead based on a static set of persona-related text information. Therefore, these prior works cannot model the effect of dynamic attributes, such as the changing relationship closeness between a pair of agents. Furthermore, Humanoid Agents can simultaneously consider multiple aspects (e.g. basic needs fulfillment, emotion and relationship closeness) in determining appropriate dialogue responses, as humans do while prior works only consider one relevant aspect at a time.

## 3 Humanoid Agents

Illustrated in Fig. 2, the architecture of Humanoid Agents is based on Generative Agents with improvement to Agent Initialization, Activity Planning and Dialogue Generation in order to account for System 1 thinking processes. Akin to Generative Agents, ChatGPT 3.5 is used for all generations, but support for other language models is planned for future development.

### 3.1 Agent Initialization

Similar to Park et al. (2023), we initialize each agent with a name, age, an example day plan, a list of sentences describing the agent (e.g. John Lin is a pharmacy shopkeeper at the Willow Market and Pharmacy who loves to help people) and a few of their personality traits (e.g. friendly, kind).

In addition, Humanoid Agents have their default emotion set to neutral out of 7 possible emotions: angry, sad, afraid, surprise, happy, neutral and disgusted (Ekman, 1992). Each of their basic needs (fullness, fun, health, social and energy) is set to a mid level (i.e 5 out of 10 where 0 is not meeting their need at all and 10 is fully satisfying that need) except energy which is set to full (i.e. 10 out of 10), as we would expect at the start of an agent’s day. Finally, we set social relationships for each agent with other agents, with each social relationship having a closeness field set to an integer value between 0 and 30 where below 5 is distant, 5 to 9 is rather close, 10 to 14 is close and 15 and above

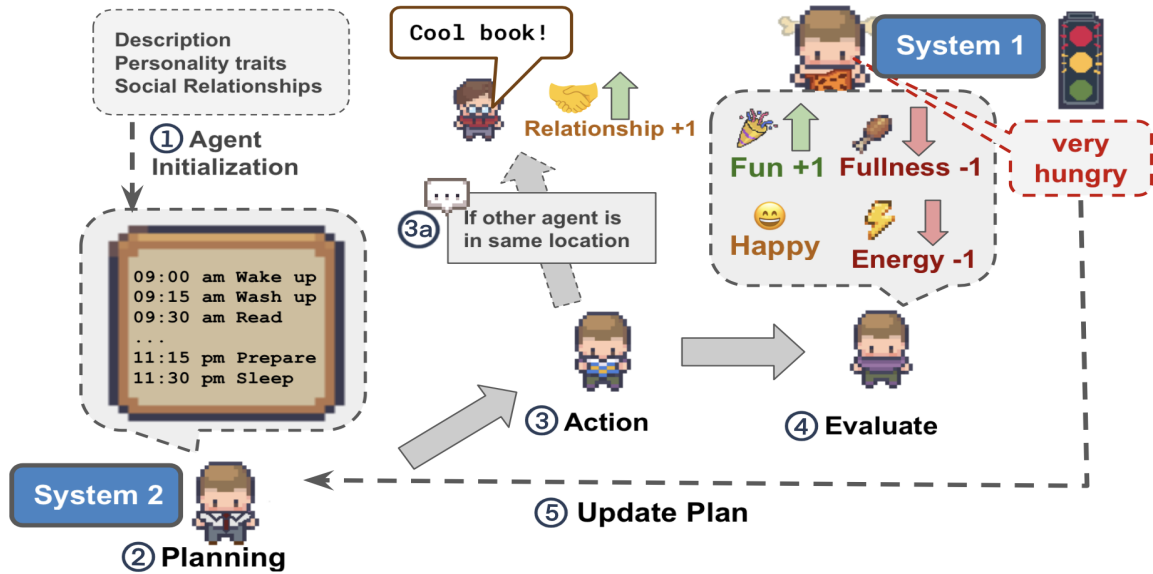


Figure 2: Architecture of Humanoid Agents. **Step 1:** Agent is initialized based on user-provided seed information. (Details in Section 3.1) **Step 2:** Agent plans their day. **Step 3:** Agent takes an action based on their plan. **Step 4:** Agent evaluates if action taken changes their basic needs status and emotion. **Step 5:** Agent can update their future plan based on the satisfaction of their basic needs and emotion. (Section 3.2) **Step 3a:** Agent can converse with another agent if in the same location, which can affect the closeness of their relationship. (Section 3.3)

is very close. Unless otherwise specified, the initial closeness value is set to 5 (rather close) to allow relationship closeness to develop over time.

### 3.2 Activity Planning

Briefly, we follow Park et al. (2023) to determine the activities of an agent by first planning out an entire day at the start of the day, using their example day plan, personality traits and description. Then, the day plan is recursively decomposed into plans at 1 hour intervals and then 15 minute intervals to improve the logical consistency of activities over time. This plan can be updated based on observations of their environment. Every 15 minutes, agents carry out an activity in their plan, with their location determined based on their previous location, the nature of their activity and available locations in their world.

We supplement this activity planning by enabling Humanoid Agents to update plans in response to changes in their internal states (*i.e.* emotions and basic needs). If an agent’s emotion is not neutral or if any basic needs is unmet (*i.e.* 3 or below out of 10), we format the agent’s internal state into natural language based on a modifier (3 for slightly, 2 for <no-modifier>, 1 for very and 0 for extremely) and an adjective (hungry for fullness, bored for fun, unwell for health, tired for energy and lonely for social). We use this for-

matted internal state as well as their original plan from the current time onward to determine if they should change their plan, and if so, how they should change it in 1 sentence. If a plan change is given, we use the suggested change and their original plan from the current time onward to generate an updated plan. For instance, if the agent is very hungry currently but only plans to have a full meal in 3 hours, the agent can have a snack while continuing their current activity, similar to how people might act. After an agent engages in an activity, the agent evaluates if doing so changes its emotion and satisfies any of its basic needs, increasing the corresponding basic need status by one when applicable. Otherwise, each basic need has a set likelihood to decrease by one over time, similar to how people naturally get hungry when they are not eating.

### 3.3 Dialogue Generation

As with Generative Agents (Park et al., 2023), Humanoid Agents in the same location can decide if they want to engage in a conversation. An agent uses a variety of factors (*e.g.* its personality traits, core characteristics, current daily occupation as well as feelings towards their progress in life, the activity they are engaging in as well as the other’s agent’s activity) to determine if they want to have a dialogue with the other agent and if so, the topic they wish to talk about. The agent can then de-

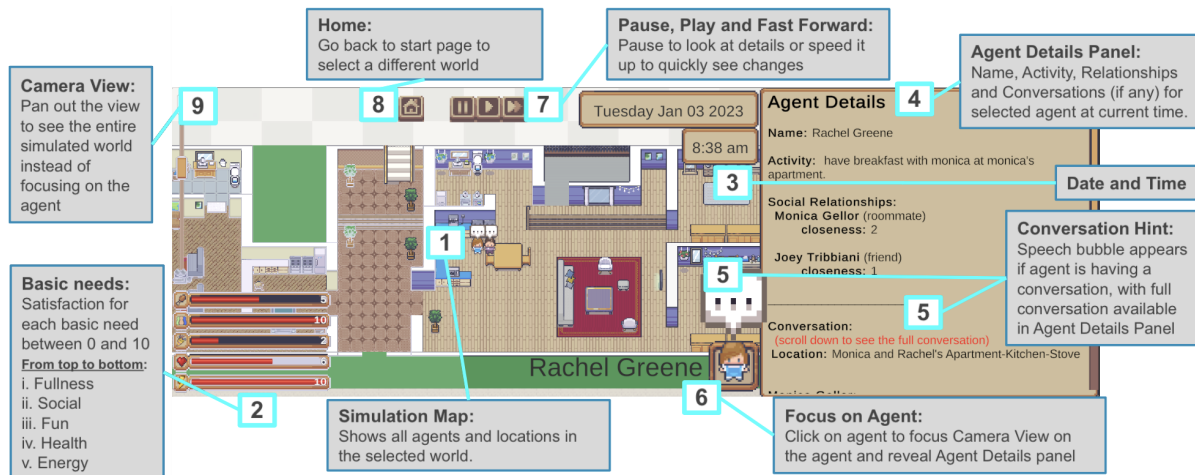


Figure 3: Unity WebGL Game Interface for visualizing Humanoid Agents situated in their world.

termine what they say, based on the decided topic. The other agent can use this conversational history in addition to the same set of factors considered by the first agent, to determine if and how they would reply. This process alternates between the two agents until one of them decides not to respond. To mitigate the likelihood that the total length of factors considered as well as conversational history exceeding the maximum context window of 4096 tokens, we limit the max number of turns to 10.

Humanoid Agents can also make use of its emotion and basic needs status as well as the relationship closeness with the other agent converted into a natural language description (e.g. John Lin is feeling close to Eddy Lin) to determine if and how they want to engage with the other agent. At the end of the dialogue, each agent will use the conversation history to determine if they enjoyed the conversation. If so, their closeness to the other agent will increase by one, otherwise, their closeness will decrease by one. We allow the relationship closeness to gradually change (*i.e.* by one out of five points required to qualitatively change their closeness from rather close to close) to reflect how relationships between humans develop over time. Furthermore, we use the conversation history to determine if the emotions of the agents are affected by the conversation.

## 4 Platform Implementation

In this section, we introduce how we build worlds and model the passage of time on our platform. Then, we describe how our Unity WebGL Game Interface and Interactive Analytics Dashboard can help users to visualize agent statuses.

### 4.1 Worlds

We use Lin’s Family World and two agents (John Lin and Eddy Lin) whose information can mostly be extracted from Park et al. (2023). Lin’s Family World also comes with a description of the locations within it, such as the Market and Pharmacy, College and Lin’s Family House with bedrooms for Eddy and John respectively. In addition, we create two other worlds - Friends and Big Bang Theory - and plan to support additional user-defined worlds on our platform. We choose these worlds because they contain personalities that are well-known to many (*e.g.* Sheldon, Leonard and Penny from Big Bang Theory; Rachel, Joey and Monica from Friends). Briefly, we used character information from <https://the-big-bang-theory.com/> and <https://friends.fandom.com/wiki/> and prompted ChatGPT 3.5 to generate descriptions and an example day plan for each character, based on information only available before Season 1 Episode 1 of each series. For agents in the Big Bang Theory and Friends, the relationship closeness is initially set to between 1 and 5 to reflect how acquainted agents are at the start of the series.

### 4.2 Time

We simulate each world for two weekdays at 15-minute intervals between 6:00 am to 12:00 midnight. Across various basic needs, fullness and health are expected to reduce by 1 every 5 hours while social and fun are expected to be reduced by 4 and energy by 5 in the same time. These values are set based on the rate that agents satisfy these basic needs through their activities, such that basic needs can adequately influence agent behavior.

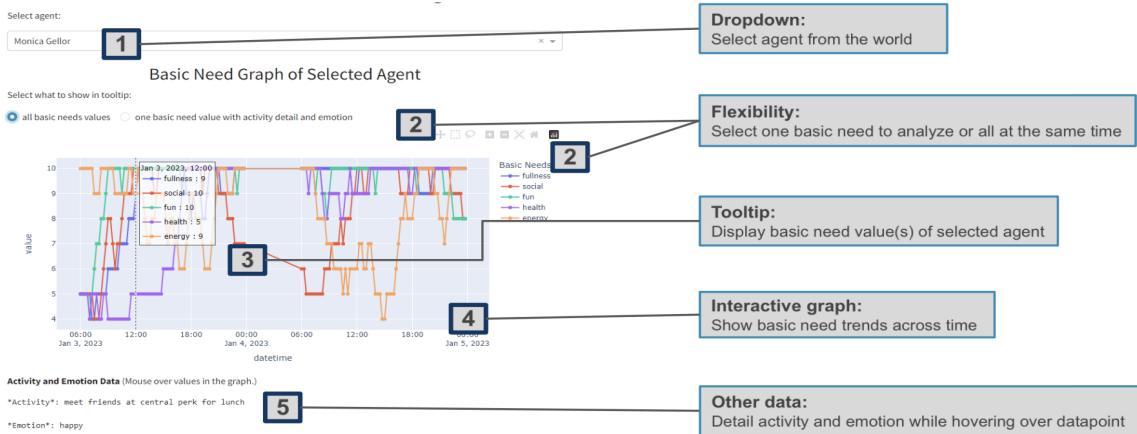


Figure 4: Interactive Analytics Dashboard for visualizing basic needs satisfaction of Humanoid Agent over time.

### 4.3 Unity WebGL Game Interface

We create a sandbox HTML game environment using Unity WebGL game engine<sup>1</sup> to visualize our humanoid agents in their worlds. Users can select from one of the three worlds to see the agent status (e.g. basic needs, emotion, activity, social relationships and conversation) and location at each time-step. Our game interface ingests JSON-structured files from our simulated worlds and transforms them into animations. The game interface also supports various functionalities for user interactions, as detailed in Fig. 3.

### 4.4 Interactive Analytics Dashboard

Users can visualize the status of various humanoid agents over time using our interactive web dashboard built with Plotly Dash.<sup>2</sup> Users can select an agent from a world to view time-series graphs relating to the satisfaction of their various basic needs, as shown in Fig. 4. Alternatively, users can view how the social relationships of agents develop over time, in terms of their closeness to one another. Users can further interact with the dashboard to see details of agents such as their emotions and activities while hovering over each point on the time-series trend-line. This can be helpful for researchers such as computational social scientists who are interested in understanding how various aspects of System 1 fluctuate over time. Other aspects of System 1 can also be visualized as they are supported by future Humanoid Agents.

<sup>1</sup><https://docs.unity3d.com/Manual/webgl-building.html>

<sup>2</sup><https://github.com/plotly/dash>

## 5 Experiments

We first investigate the effectiveness of Humanoid Agents in evaluating the effects of activities and conversations have on various aspects of System 1, by comparing its predictions with human annotations. Then, we study the effects that System 1 attributes have on activities and conversations. Due to page length limitations, we only present the effects of basic needs on activities, and leave the effects of emotions and relationship closeness to Appendices A.1 and A.2 respectively.

### 5.1 Comparison with Human Annotations

To understand how well Humanoid Agents are able to predict 1. whether activities satisfy various basic needs; 2. emotions expressed in activities and; 3. if dialogues bring two agents closer, we compare our system’s predictions with human annotations. Three volunteer human annotators labelled the simulation of 1 day in Lin’s Family World using the same instructions given to the language model within our system (details in Appendix A.3). 144 activities are annotated independently by each annotator for emotions and each basic need, while there are 30 annotations for user-conversation pairs. We then take majority vote across all annotators and calculate micro-F1 between the majority vote and the system predictions.

Table 1 shows good inter-rater reliability (Fleiss’  $\kappa \geq 0.556$ ) across all basic needs, emotion and relationship closeness. We find that our system is able to perform well (F1  $\geq 0.84$ ) on classifying if an activity increases fullness and energy; the emotion expressed in an activity and; whether a conversation brings agents closer to one another. However, it slightly struggles in classifying whether activi-

|                    | Fleiss' $\kappa$ | F1    |
|--------------------|------------------|-------|
| <b>Basic Needs</b> |                  |       |
| fullness           | 0.972            | 0.972 |
| social             | 0.833            | 0.743 |
| fun                | 0.806            | 0.66  |
| health             | 0.917            | 0.694 |
| energy             | 0.88             | 0.861 |
| <b>Emotion</b>     |                  |       |
|                    | 0.823            | 0.84  |
| <b>Closeness</b>   |                  |       |
|                    | 0.556            | 0.9   |

Table 1: Human evaluation on the capability of our system to predict if activities satisfy various basic needs, emotions expressed in activities and whether a conversation brings two agents closer.

ties satisfy basic needs of fun, health and social. One possible cause is that our system substantially over-predicts the number of activities contributing to these basic needs (health 34% of predicted activities vs 4.9% human-annotated activities, fun 44.4% vs 10.4% and social 47.2% vs 24.3%). More specifically, our system mispredicts activities relating to medication for others (since John Lin works at a Pharmacy) as contributing to the agent’s own physical health; common activities relating to agents’ occupation as enjoyable (*e.g.* receive feedback from professor or help regular customers with their medication needs); activities where the presence of other agents is unspecified as social (*e.g.* organize the counter and display areas or check the inventory and replenish any low stock). The use of language models with greater capabilities in commonsense understanding can potentially mitigate this issue (Bosselut et al., 2019; OpenAI, 2023).

## 5.2 Effects of Basic Needs on Activities

Given that humanoid agents operate as a dynamic system with many components, it can be challenging to isolate the effect of each basic need on agent activities. To investigate the contribution of each basic need, we simulate worlds with agents having one basic need initially set to zero, making agents extremely hungry, loneliness, tired, unwell or bored at the start of the day. We study the amount of time in one simulated day that agents spent performing activities to satisfy that basic need (*e.g.* eating food to overcome hunger or socializing to alleviate loneliness). Then, we compare it with the amount of time that agents spent performing such activities in normal settings (where every basic need is set to 5 and energy is set to 10), to calculate the percentage increase in time spent on fulfilling each basic need for our test conditions.

| Basic Need | % change in time spent satisfying basic need |     |         |     |     |                 |     |     |  | Mean $\mu$ |
|------------|--|-----|---------|-----|-----|-----------------|-----|-----|--|------------|
|            | Lin’s Family                                 |     | Friends |     |     | Big Bang Theory |     |     |  |            |
|            | JL   | EL  | MG      | RG  | JT  | SC              | LH  | P   |  |            |
| health     | 211  | 57  | 53      | 210 | 240 | 207             | 207 | 60  |  | 156        |
| fullness   | 62   | 36  | 62      | -45 | 50  | 112             | 0   | 0   |  | 35         |
| fun        | -10  | -22 | 6       | 12  | -24 | -12             | 3   | -26 |  | -9         |
| social     | 10   | 31  | 32      | -15 | -13 | 4               | 17  | 34  |  | 12         |
| energy     | 38   | 96  | 47      | 81  | 80  | -39             | 153 | -5  |  | 56         |

Table 2: Effects of setting each basic need status to zero on percentage change in time spent on activities fulfilling those needs, relative to normal settings. *Initials of characters* - JL: John Lin, EL: Eddy Lin, MG: Monica Gellor, RG: Rachel Greene; JT: Joey Tribbiani, SC: Sheldon Cooper, LH: Leonard Hofstadter, P: Penny

As shown in Table 2, humanoid agents adapt their activities most when the basic needs of health (156%), energy (56%) and fullness (35%) are initialized to zero. This supports their importance, as Maslow (1943) grouped them into low-level physiological and safety needs that people need to satisfy before fulfilling other needs. In response to these conditions, agents typically look for medical support, get more rest or seek more food. On the other hand, when agents feel lonely due to the lack of social interactions, they only slightly adapt their behavior (+12%) to interact more with other agents.

Another contributory factor to the small changes for both social and fun is that agents in normal settings already spend a large amount of time doing activities that contribute to these basic needs: on average, they spend 11 (out of 18 simulated hours) doing something they enjoy, 8.75 hours on social interactions and only 5.75 hours for resting and 2.75 hours each for eating and doing something that improves their health.<sup>3</sup> This means that the effects of setting either fun or social to zero initially dissipates very earlier in the day, giving way to other priorities including work obligations, such as Penny working at the Cheesecake Factory.

## 6 Conclusion

We propose Humanoid Agents, a platform for human-like simulations of Generative Agents guided by System 1 processing including Basic Needs (*e.g.* hunger, health and energy), Emotion and Closeness in Relationships. Our platform also powers the immersive visualization of agents using our Unity WebGL Game Interface and an Interactive Analytics Dashboard.

<sup>3</sup>Activities such as meeting friends for lunch can be enjoyable, social and filling at the same time, so the total time across all basic needs can be more than 18 hours.



## Limitations

**Multiparty Dialogue:** Our system currently only supports dialogue between two agents, even if there are more than two agents in the same location. We aim to support multi-party conversation in the future.

**Synchronization of activities between agents:** Activity planning is done by each agent independently and not forcibly synchronized with each other. For instance, John Lin can plan to watch a movie with Eddy Lin at 8:00pm but Eddy Lin can be calling his friends at 8:00pm. We plan to synchronize activities between agents in the future, by prompting one or more agents to update their plan if their plans are not coherent with one another.

**Variability in Natural Decline of Agent Basic Needs:** The rates of decrease for various basic needs is the same for all agents in our current implementation. We plan to allow these rates to be customized for each character to account for individual differences (e.g. extroverts have faster decline of social fulfillment and people who get hungry easily can have fullness reduced more quickly).

## Ethics Statement

**Broader impacts:** Our system will help researchers such as computational social scientists to be better able to simulate human behavior in-silico before doing further studies in the real-world. This is particularly helpful if real-world studies are difficult or costly.

**Risk:** While our system allows simulated agents to behave more like humans, it is not perfect and should not be treated as so. Users of our simulation platform must be informed that they are working with a simulation that does not perfectly reflect human behavior in the real world, so that they do not overly trust the results of the simulation.

## Acknowledgements

We thank the anonymous reviewers for their helpful comments.

## References

Sieun An, Li-Jun Ji, Michael Marks, and Zhiyong Zhang. 2017. [Two sides of emotion: Exploring positivity and negativity in six basic emotions across cultures](#). *Frontiers in Psychology*, 8.

Kira S. Birditt, Laura M. Miller, Karen L. Fingerma, and Eva S. Lefkowitz. 2009. [Tensions in the parent and adult child relationship: Links to solidarity and ambivalence](#). *Psychology and Aging*, 24(2):287–295.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. [COMET: Commonsense transformers for automatic knowledge graph construction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Harrison Chase. 2022. [LangChain](#).

R.I.M. Dunbar. 2009. [The social brain hypothesis and its implications for social evolution](#). *Annals of Human Biology*, 36(5):562–572.

Paul Ekman. 1992. [An argument for basic emotions](#). *Cognition and Emotion*, 6(3-4):169–200.

Barry G. Ginsberg. 1996. [Together in group therapy: Fathers and their adolescent sons](#). In *Men in groups: Insights, interventions, and psychoeducational work.*, pages 269–282. American Psychological Association.

Michael L. Hecht. 1984. [Satisfying communication and relationship labels: Intimacy and length of relationship as perceptual frames of naturalistic conversations](#). *Western Journal of Speech Communication*, 48(3):201–216.

Siddharth Karamcheti Saachi Jain Samuel Humeau Emily Dinan Tim Rocktäschel Douwe Kiela Arthur Szlam Jason Weston Jack Urbanek, Angela Fan. 2019. [Learning to speak and act in a fantasy text adventure game](#).

Liwei Jiang, Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jenny Liang, Jesse Dodge, Keisuke Sakaguchi, Maxwell Forbes, Jon Borhardt, Saadia Gabriel, Yulia Tsvetkov, Oren Etzioni, Maarten Sap, Regina Rini, and Yejin Choi. 2022. [Can machines learn morality? the delphi experiment](#).

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. [Camel: Communicative agents for "mind" exploration of large scale language model society](#).

Fangyu Liu, Emanuele Bugliarello, Edoardo Maria Ponti, Siva Reddy, Nigel Collier, and Desmond Elliott. 2021. [Visually grounded reasoning across languages and cultures](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10467–10485, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

- Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M. Dai, Diyi Yang, and Soroush Vosoughi. 2023. [Training socially aligned language models in simulated human society](#).
- A. H. Maslow. 1943. [A theory of human motivation](#). *Psychological Review*, 50:370–396.
- Rowland S. Miller. 2012. *Intimate relationships*. McGraw-Hill.
- Anton J. Nderhof. 1985. [Methods of coping with social desirability bias: A review](#). *European Journal of Social Psychology*, 15(3):263–280.
- OpenAI. 2023. [Gpt-4 technical report](#).
- OpenBMB. 2023. [AgentVerse](#).
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#).
- Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. [Towards empathetic open-domain conversation models: A new benchmark and dataset](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics.
- Sam G.B. Roberts and Robin I.M. Dunbar. 2011. [The costs of family and friends: an 18-month longitudinal study of relationship maintenance and decay](#). *Evolution and Human Behavior*, 32(3):186–197.
- Richard M. Ryan and Edward L. Deci. 2001. [On happiness and human potentials: A review of research on hedonic and eudaimonic well-being](#). *Annual Review of Psychology*, 52(1):141–166.
- Ashish Sharma, Adam Miner, David Atkins, and Tim Althoff. 2020. [A computational approach to understanding empathy expressed in text-based mental health support](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5263–5276, Online. Association for Computational Linguistics.
- Gravitas Significant. 2023. [Auto GPT](#).
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. [Voyager: An open-ended embodied agent with large language models](#).
- Zhilin Wang and Pablo E. Torres. 2022. [How to be helpful on online support forums?](#) In *Proceedings of the 4th Workshop of Narrative Understanding (WNU2022)*, pages 20–28, Seattle, United States. Association for Computational Linguistics.
- Zhilin Wang, Xuhui Zhou, Rik Koncel-Kedziorski, Alex Marin, and Fei Xia. 2022. [Extracting and inferring personal attributes from dialogue](#). In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 58–69, Dublin, Ireland. Association for Computational Linguistics.
- Xiaodong Wu, Weizhe Lin, Zhilin Wang, and Elena Rastorgueva. 2020. [Author2vec: A framework for generating user embedding](#).
- Nakajima Yohei. 2022. [BabyAGI](#).
- Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. [Personalizing dialogue agents: I have a dog, do you have pets too?](#)
- W.-X. Zhou, D. Sornette, R. A. Hill, and R. I. M. Dunbar. 2005. [Discrete hierarchical organization of social group sizes](#). *Proceedings of the Royal Society B: Biological Sciences*, 272(1561):439–444.

## A Appendices

### A.1 Effects of Emotions on Activities

To understand the effects of emotions on the activity of agents, we apply a similar approach as we do with basic needs in setting the initial emotion to one other than neutral. Unlike basic needs however, emotion does not contain gradation and therefore changes much more quickly than basic needs statuses. An agent who wakes up sad can become neutral right after washing up, severely limiting the effects of setting the initial emotion on activities over the rest of the day. To overcome this issue, we disable the updating of emotions based on activities and dialogues for agents in this part of the study, making the agents resemble humans who are stuck in a particular emotion over the entire day.

We study the number of times in one simulated day to which agents perform activities (at 15-min intervals) that express each emotion. For instance, when agents are *angry*, they go for a run to release anger; when *sad*, they seek support from a trusted friend; when *disgusted*, they practice deep breathing exercises and meditation techniques; and when *surprised*, they take time to process and reflect on the surprising findings. Then, we calculate the difference to the number of times that agents perform such activities in normal settings (initialized with neutral and allowed to update based on activities and dialogues). Because agents in normal settings typically do not perform activities that express sadness, anger, fear, disgust or surprise, we report the increase in number of activities expressing those emotions, compared to agents in normal settings.

As shown in Table 3, anger influences agent behavior most (+15 activities) followed by sadness

| Emotion   | change in no. of activities expressing emotion |    |         |    |     |                 |    |    |           |
|-----------|--|----|---------|----|-----|-----------------|----|----|-----------|
|           | Lin's Family                                   |    | Friends |    |     | Big Bang Theory |    |    | Mean      |
|           | JL   | EL | MG      | RG | JT  | SC              | LH | P  | $\mu$     |
| angry     | 7  | 15 | 18      | 18 | 12  | 22              | 23 | 8  | <b>15</b> |
| sad       | 6  | 22 | 8       | 8  | 15  | 9               | 8  | 6  | <u>10</u> |
| afraid    | 3  | 16 | 4       | 1  | 12  | 5               | 14 | 24 | <u>10</u> |
| disgusted | 5  | 1  | 16      | 0  | 2   | 4               | 0  | 5  | 4         |
| surprised | 0  | 1  | 0       | 0  | 0   | 3               | 4  | 2  | 1         |
| happy     | 6  | 0  | -9      | -9 | -18 | -4              | -6 | 16 | -3        |

Table 3: Effects of fixing agent emotion on the change in number of activities (in 15-minute intervals) that agents perform expressing set emotion, relative to normal settings. *Initials of characters* - JL: John Lin, EL: Eddy Lin, MG: Monica Gellor, RG: Rachel Greene; JT: Joey Tribbiani, SC: Sheldon Cooper, LH: Leonard Hofstadter, P: Penny

and fear (+10 each), then disgust (+4) and surprise (+1) and finally happiness (-2). Negative emotions seem to influence agents much more than positive emotions (*i.e.* happiness), as agents often do not plan for activities with negative emotions and hence have to significantly adjust their plans to manage their negative emotions, as with converging evidence in humans (An et al., 2017). Among the negative emotions, disgust and surprise are transient emotions that humans and agents do not typically experience for a long time, therefore limiting their influence compared to persistent ones (sadness, fear and anger) that can affect their activity over the entire day. It is interesting to observe that agents do slightly fewer (-2) activities to make themselves happy when they are already happy, possibly because feeling happy empowers them to pursue activities serving other longer term goals, even if these activities are not immediately joy-inducing (Ryan and Deci, 2001). For instance, when emotion is set at happy, Joey Tribbiani devotes more time to practicing his craft as an actor and Sheldon Cooper spends more time doing research.

## A.2 Effects of Closeness on Dialogues

We investigate the effects of initial relationship closeness on dialogue between two agents. We set the closeness between all pairs of characters to either 0 (distant), 5 (rather close), 10 (close) or 15 (very close). We consider only the first five conversations between all agents in each simulation, in order to ensure that the closeness values between two agents are kept at the initial set level. We report the effects on the mean number of turns in each conversation as well as the proportion of turns that our system rates to have positive sentiments, since closeness between agents can influence their

| Closeness    | Mean turns  |            |            | % positive sentiment |            |            |
|--------------|-------------|------------|------------|----------------------|------------|------------|
|              | LF          | F          | BBT        | LF                   | F          | BBT        |
| Distant      | 5.4         | 6.8        | 8.33       | 88.9                 | <b>100</b> | <b>100</b> |
| Rather Close | <b>7.25</b> | 7.8        | 8.8        | 93.1                 | 97.4       | <b>100</b> |
| Close        | 6.2         | <b>8.2</b> | <b>9.8</b> | <b>96.8</b>          | 97.5       | 98.0       |
| Very Close   | 6.0         | 7.2        | 8.0        | 90.0                 | 97.2       | 97.5       |

Table 4: Effects of closeness between agents on the mean number of turns in their conversations and the proportion of turns with positive sentiments. *Initials to World* - LF: Lin's Family, F: Friends, BBT: Big Bang Theory

willingness to express positive/negative sentiment in their conversations with one another.

As shown in Table 4, the mean number of conversation turns typically follow an inverse U shape as closeness increases. Agents talk less when they are distant, more when they get closer, but then tapers off when they get very close. This is supported by converging evidence in human conversations where we feel less of a need to engage in politeness talk when we feel very close to others (Hecht, 1984). In Lin's Family (LF), the turning point is at Rather Close while in Friends (F) and Big Bang Theory (BBT), it is at Close. This is likely between the two agents in LF have a father-son relationship where agents feel comfortable communicating less at a lower level of closeness without straining the relationship (Ginsberg, 1996) while in F and BBT, agents are friends and neighbours who require more active communication to maintain as non-kin relationships (Roberts and Dunbar, 2011).

Proportion of conversational turns with positive sentiment generally goes down in Table 4 when closeness is higher, akin to how people feel less of a need to praise others in order to be liked when very close to others (Miller, 2012). For instance, when Joey feels distant from Monica, he says 'Hey Monica, I saw you having a blast at that restaurant with your friends! The food looked incredible. What's the secret to finding such amazing places to eat? I could use some recommendations for my next date night.' When he feels very close, he instead says 'Hey Monica! I saw you having lunch at that nearby restaurant. How was the food? I'm always on the lookout for new culinary trends to try out in my cooking. Any recommendations or standout dishes you enjoyed?'

In LF, the proportion of turns with positive senti-

ment is also lower (88.9%) when agents feel distant from each other. This is possibly because when a father-child pair feel distant from each other, they can be more likely to argue with each other out of dissatisfaction with the strained relationship (Birditt et al., 2009). For instance, after dinner, Eddy Lin says to John Lin ‘Hey John, I noticed you had someone else clean up the dinner dishes tonight. Everything okay? Is there a reason for that?’ when he feels distant but says ‘I noticed how much you enjoyed that cup of tea while we were talking and bonding, and it made me appreciate our connection even more.’ when he feels very close. On the other hand, F/BBT agents entirely refrain from demonstrating negative sentiments when they are distant from their acquaintances/neighbours to maintain a positive social image, similar to humans (Nederhof, 1985). Overall, this suggests that the effect of relationship closeness on agents’ conversations is substantially moderated by their relationship type, as supported by converging evidence from human conversations.

### A.3 Templates for Evaluating Basic Needs, Emotion, Closeness and Sentiment

**Basic Needs** Does the activity {activity} involve {satisfaction-action}? Please respond only with either yes or no.

where satisfaction-action for each basic need is based on Table 5

**Emotion** In the following activity {activity}, what emotion is expressed? Please respond only with one word from this list [“neutral”, “disgusted”, “afraid”, “sad”, “surprised”, “happy”, “angry”].

**Closeness** Given this conversation {conversation}, did {name} enjoy the conversation? Please respond with either yes or no.

**Sentiment** In the following utterance {utterance}, is the sentiment positive? Please respond only with either yes or no.

| Basic Need | Satisfaction-action                                     |
|------------|---|
| fullness   | eating food   |
| social     | interacting with other people                           |
| fun        | doing something enjoyable                               |
| health     | doing something that improves their own physical health |
| energy     | resting or having a break                               |

Table 5: Satisfaction-action for each basic need

# TP-Detector: Detecting Turning Points in the Engineering Process of Large-scale Projects

Qi Wu<sup>1</sup>, Wenhan Chao<sup>1</sup>, Xian Zhou<sup>2</sup> and Zhunchen Luo<sup>2\*</sup>

<sup>1</sup> School of Computer Science and Engineering, Beihang University  
{wuqismile, chaowenhan}@buaa.edu.cn

<sup>2</sup> Information Research Center of Military Science, PLA Academy of Military Science  
zhouxian@sjtu.edu.cn, zhunchenluo@gmail.com

## Abstract

This paper introduces a novel task of detecting turning points in the engineering process of large-scale projects, wherein the turning points signify significant transitions occurring between phases. Given the complexities involving diverse critical events and limited comprehension in individual news reports, we approach the problem by treating the sequence of related news streams as a window with multiple instances. To capture the evolution of changes effectively, we adopt a deep Multiple Instance Learning (MIL) framework and employ the multiple instance ranking loss to discern the transition patterns exhibited in the turning point window. To facilitate comprehensive evaluation of the task, we curate a dataset comprising 80 large-scale projects. Extensive experiments consistently demonstrate the effectiveness of our proposed approach on the constructed dataset compared to baseline methods. We deployed the proposed model <sup>1</sup> and provided a demonstration video<sup>2</sup> to illustrate its functionality. The code and dataset are available on GitHub<sup>3</sup>.

## 1 Introduction

Large-scale projects are intricate and extensive endeavors requiring substantial resources, effort, and coordination to achieve specific objectives, often involving multiple stakeholders and phases with a significant impact on organizations, communities, and society. They encompass diverse fields, such as aerospace engineering, water resources facilities, and transportation infrastructure, and hold paramount importance in driving economic growth, enhancing infrastructure, and addressing societal needs, while also fostering innovation and sustain-

ability. Typically, their successful execution requires careful planning, collaboration, and a long-term vision to maximize their positive impact on communities and society at large.

In order to tackle the intricacies and difficulties of carrying out large-scale projects, the engineering process employs a systematic and structured approach to design, plan, and execute complex endeavors efficiently and effectively (Martin, 2000; Gilb, 2005). The life cycle of large-scale projects consists of several phases that cover the entire journey from initiation to closure (Beitz et al., 1996; Bennett, 2003) and each phase dedicates to accomplishing different objectives. Figure 1 depicts the seven phases involved in NASA’s Insight mission engineering process, wherein each phase comprises a series of subtasks or key events represented by the gray diamonds, which can occur simultaneously or have interdependencies.

During the engineering process of large-scale projects, there are significant moments that deserve attention, such as when the project reaches a new milestone, e.g., transitioning from the conceptual phase to on-ground implementation on the ground (as shown in the green diamond *h* of Figure 1). The moments or events which bring about critical changes in direction, course, or outcome are referred to *turning points* in this paper, signifying transitions occurring between adjacent phases. The identification of these turning points provides stakeholders, government agencies, and the general public with valuable insights, empowering them to navigate challenges, capitalize on opportunities, and effectively adjust their strategies in response to changing circumstances. For example, an analyst can assess Boeing’s development status and offer design suggestions for products at Airbus.

However, detecting turning points in the engineering processes of large-scale projects, particularly ongoing projects, is a non-trivial undertaking. First, limited public data sharing and potential lack

\*Corresponding Author

<sup>1</sup><http://43.138.60.114:7080/>

<sup>2</sup><https://youtu.be/FH3av84I-Kg>

<sup>3</sup><https://github.com/smile577/tpd>

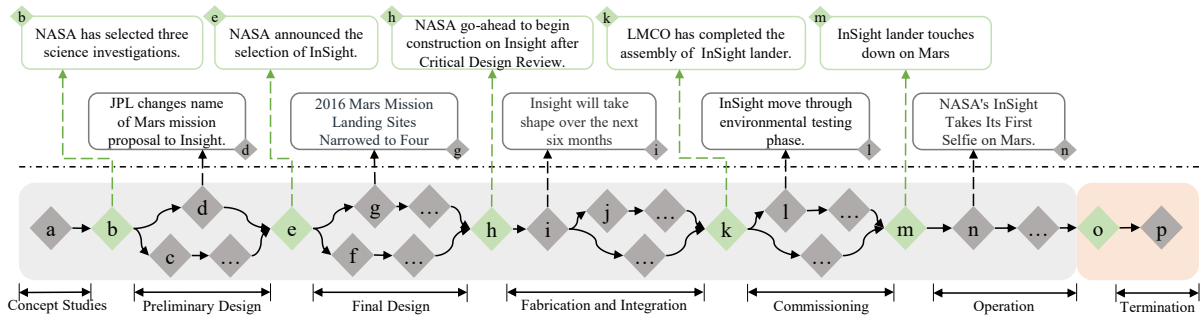


Figure 1: Illustration of the engineering process of NASA’s InSight mission. The gray diamonds represent subtasks or events within a phase, whereas the green diamonds represent turning points that occur during the transition between phases.

of standardized protocols may obstruct access for those not directly engaged. Second, listing all possible turning point events for various projects is challenging. Additionally, the turning points may encompass multiple critical events, and the same event could be part of a phase or being in a transition under different circumstances. Nevertheless, publicly available news serves as a conveniently obtainable and reasonably trustworthy data source. In light of this, we’ve discovered that discerning the occurrence of a turning point within a series of interconnected news articles can be accomplished by comparing and analyzing the significant events. To accomplish this, we view a sequence of related news streams as a window and resort to the Multiple Instance Learning (MIL) (Andrews et al., 2002; Sultani et al., 2018) framework to obtain window-level labels, indicating the presence or absence of a turning point within the window. In order to avoid biased understanding of individual reports, the instances of a window comprise several news to form a relatively comprehensive grasp of the events that occurred.

Following the insights, this paper delves into the intricate task of detecting turning points in the engineering process of large-scale projects by utilizing multiple instance learning techniques. To facilitate this study, we collect 80 large-scale projects to construct the turning point detection dataset. We structure the sequence of related news streams into a window with multiple instances through a deep MIL framework and identify turning points via convolutional transformer encoder (Dosovitskiy et al., 2020; Li et al., 2022). We then employ the multiple instance ranking loss to push the positive instances and negative instances far apart in terms of the extent of change or shift that occurs between phases. Additionally, we develop a website for detection and visualization with our deployed model, provid-

ing users with transition timeline and news related to single or multiple large-scale projects.

## 2 Related Work

**Turning Point Detection** In the large-scale projects, specific methods for detecting turning points have not yet been established. However, similar concepts exist in other domains such as time series change-point detection, as well as turning point detection in video and text data. In the field of time series data change-point detection (Aminikhanghahi and Cook, 2017; Truong et al., 2020), the focus is primarily on identifying fluctuations in time series data, such as those in financial stock markets (Grillenzoni, 2012; Tang et al., 2019) and weather temperatures (Banesh et al., 2019). These methods commonly detect changes based on fluctuations along specific dimensions in low-dimensional spaces. However, these techniques may not be directly applicable to high-dimensional data such as video and text. In case of video sequences, Chang et al. (Liu et al., 2019a) introduced the use of the Two Clocks theory (Lotker, 2016) to detect a key event in narrative works, aiming to identify multiple turning points in cartoon movie stories. In the text sequences, Papalampidi et al. (Papalampidi et al., 2019) proposed the task of identifying turning points in movie screenplays to analyze narrative structures. They defined turning points in screenplays and developed an end-to-end neural network model for recognition.

**Multiple Instance Learning** Multiple Instance Learning (MIL) is a form of weakly supervised learning where training instances are organized into sets, called bags (Maron and Lozano-Pérez, 1997; Herrera et al., 2016). Only the label for the bag is provided. Due to this characteristic, MIL has found extensive applications in domains with large amounts of weakly labeled data (Quelleg et al.,

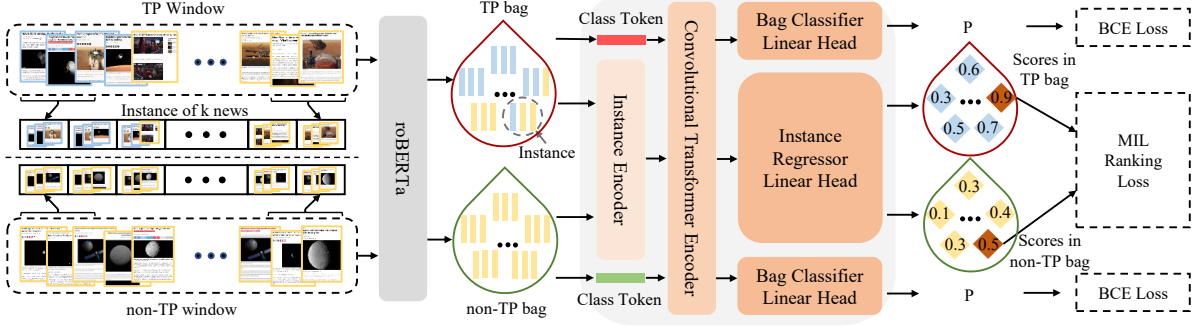


Figure 2: An overview of multiple instance learning framework for turning point detection of large-scale projects.

2017; Tian et al., 2020), such as video classification (Ding et al., 2013; Sultani et al., 2018; Li et al., 2022), image classification (Sudharshan et al., 2019; Li et al., 2021; Yang et al., 2023), and text classification (He and Wang, 2009; Liu et al., 2018), among others. In the context of weakly supervised video classification, typically only video-level category labels are provided. Sultani et al. (Sultani et al., 2018) proposed the MIL ranking model. They utilized the MIL ranking model to compute the highest-scoring instance within the bag for video classification. Li et al. (Li et al., 2022) introduced a MIL ranking model based on Transformer for video classification. In the realm of text classification, He et al. (He and Wang, 2009) proposed a KNN algorithm-based multi-instance Chinese text classifier. Liu et al. (Liu et al., 2018) introduced Selective Multi-Instance Transfer Learning to address the issue of knowledge-safe transfer in multi-instance learning for text classification.

### 3 TP-Detector

#### 3.1 Task Definition

Our task is defined as follows: We want to detect TP in an input news stream, where TP is typically associated with changes in sequences. To detect changes of sequences within a specific range, we partition the news stream into windows and check for the presence of TP within these windows. Given a window  $W$  on the news stream, window  $W$  contains multiple news texts, denoted as  $W = \{x_1, x_2, \dots, x_n\}$ . The window  $W$  has two classes: TP window  $W_{tp}$  and non-TP window  $W_{ntp}$ . If there is a TP within the window, meaning that events within the window across different phases, it is classified as a TP window; otherwise, it's a non-TP window. The outputs include  $Y$  and  $Y_{tp}$ .  $Y$  represents whether window  $W$  is a TP win-

dow. If  $Y = 1$ , it's a TP window; if  $Y = 0$ , it means it's not.  $Y_{tp}$  represents the evidence within  $W$  that is most likely to be a TP, and it's only output when  $Y = 1$ .

#### 3.2 Model Overview

We propose a multi-instance learning model, as shown in Figure 2. The model takes two windows from one news stream as input, namely TP window and non-TP window. It starts by employing a pre-trained language model (Liu et al., 2019b) to represent the textual features. Next, the features of continuous  $k$  news within each window are organized as instances. These instances are then processed by an Instance Encoder, which consists of multiple layers of 1D convolutions, to extract their feature representations. At this point, the window is treated as a bag containing multiple instances. Subsequently, a Transformer encoder with convolutional layers (Dosovitskiy et al., 2020; Li et al., 2022) is employed to attend to the feature representations of both the bag and the instances within the bag. This process helps the model improve its understanding of the features within the bag and its constituent instances. Finally, two linear heads assign scores to the bags and instances. The decision of whether the input window is a TP window is made based on the scores of the bags and their instances.

#### 3.3 Multiple Instance Learning

Due to the diversity of turning point events in large-scale project engineering processes, it is impractical to exhaustively enumerate all possible turning points. However, we have found that by comparing and analyzing known instances of turning points, it is possible to learn the general patterns of turning points and detect unknown turning point events. Turning point events are typically sparse

in the news streams, with the majority of information being non-TP related. To effectively discover critical turning point events, we introduce a multi-instance learning framework (Andrews et al., 2002; Sultani et al., 2018). Multi-instance learning is a weakly supervised method where the data unit is a bag. Taking binary classification as an example, a bag contains multiple instances. If at least one instance in a bag is a turning point instance, the bag is considered a turning point bag. Otherwise, it is a non-TP bag. This approach enables us to detect turning points even when they are sparsely distributed within the news streams.

We accept two windows,  $W_{tp}$  and  $W_{ntp}$ , as inputs, each containing multiple news articles. We use the pre-trained language model RoBERTa (Liu et al., 2019b) to extract feature representations for each news within the windows. The output of the feature representations for the TP window are denoted as

$$W_e = \{x_i^e | x_i^e = f_e(x_i), x_i \in W\}$$

where  $f_e$  is RoBERTa, and  $W_e$  is the output feature representations set.

We combine the continuous  $k$  feature representations from the above output into one instance. To obtain the feature representation of each instance, we use Instance Encoder to fuse the above feature representations. The Instance Encoder consists of multiple layers of 1D Convolutional and multiple layers of max-pooling. The formulation can be expressed as follows:

$$I_i = f_{ie}(x_i^e, \dots, x_{i+k-1}^e)$$

where  $f_{ie}$  is the Instance Encoder and  $I_i$  represents the feature representation of an instance.

To ensure that critical transition information is not missed due to data segmentation and to increase the information density within each bag based on the sparse nature of news stream data, we form instances using an overlapping approach. Specifically, we iterate through the window with a step of 1 news article to generate instances. Thus, we represent the input  $W$  as one bag:

$$B = \{I_i | i \in \{1, 2, \dots, n - k + 1\}\}$$

where  $B$  represents the multi-instance bag.

### 3.4 Turning Point Detection via Convolutional Transformer Encoder

To enhance the understanding of feature representations for instances within the bag, we utilized the

Convolutional Transformer Encoder (CTE) (Dosovitskiy et al., 2020; Li et al., 2022). The 1D convolution within the CTE enables information interaction among instances within the bag. Then, we employ the self-attention mechanism in the Transformer to enhance the understanding of feature representations for instances. Finally, two linear layers are used to classify the enhanced feature representations of both the bag and its internal instances.

We input the bag’s class information and the feature representations of bag instances into the CTE, which can be represented as follows:

$$B_{cte} = CTE(ClassToken || B)$$

where  $||$  represents concatenation.

The output of the CTE is then separately fed into two Linear Heads: the Instance Regressor Linear Head (IRLH) and the Bag Classifier Linear Head (BCLH) (Li et al., 2022) as follows.

$$B_{ir} = IRLH(B_{cte}[1, 2, \dots, n - k + 1])$$

$$p = BCLH(B_{cte}[0])$$

where  $B_{cte}[i]$  represents the  $i$ -th element of  $B_{cte}$ ,  $B_{ir}$  contains the scores of all instances in the bag, and  $p$  is the bag’s class prediction value.

During the prediction phase, to reduce the fluctuation of instance prediction scores, we use the bag’s class prediction score for calibration:  $B_p = B_{ir} * p$ , where  $B_p$  represents the final prediction scores of all instances in the bag. We select the maximum value in  $B_p$  as the prediction value for the bag. If the prediction value is greater than the threshold, the bag is classified as a TP bag; otherwise, it is classified as a non-TP bag.

### 3.5 Optimization via MIL Ranking Loss

To guide the multi-instance learning and achieve end-to-end TP detection, we introduce the MIL Ranking Loss (Sultani et al., 2018) to optimize the learning process of our model. This loss function helps us effectively train the model to distinguish between TP and non-TP instances within the bags.

To identify the category of a bag, we compare the input bags in such a way that the highest predicted score of instances in TP bags is greater than all instances in non-TP bags. The instance with the highest score in the non-TP bag is most similar to the TP instances in TP bags, which can lead to false positives. To distinguish between true positives and false positives, we aim to maximize the separation



between them. Therefore, the MIL Ranking Loss function can be formulated as:

$$l(B_{tp}, B_{ntp}) = \max(0, 1 - \max_{I_i \in B_{tp}} f(I_i) + \max_{I_i \in B_{ntp}} f(I_i))$$

Where  $B_{tp}$  and  $B_{ntp}$  represent the TP bags and non-TP bags, respectively, and  $f(I_i)$  denotes the final prediction score of instance  $I_i$ .

In general, TP occurs within a relatively short period; therefore, most of the instances in the window are non-TP instances. In other words, in a TP bag, only a few instance scores are close to 1, while the rest of the scores are close to 0. To address this sparsity, we introduce a sparsity constraint. Additionally, we include a binary cross-entropy loss for bag classification. Therefore, our ranking loss formula is as follows:

$$l(B_{tp}, B_{ntp}) = \max(0, 1 - \max_{I_i \in B_{tp}} f(I_i) + \max_{I_i \in B_{ntp}} f(I_i)) + \lambda_1 \sum_{I_i \in B_{tp}} f(I_i) + \lambda_2 BCE(p, Y)$$

## 4 Experiments

### 4.1 Dataset

Due to the unavailability of a dataset for our task, we collected dataset to train and evaluate our model. In order to make the development status of large-scale projects accessible to everyone interested, we aimed to collect data through publicly available channels rather than relying on proprietary sources. Given the accessibility, real-time nature, and authenticity of news, we chose to gather data from publicly available news sources as the primary data source for our dataset. Our data collection process involved several steps. Firstly, we gathered available projects as candidates. Next, we utilized the Google Search API to search for relevant news, resulting in a news stream. To facilitate the labeling process, we provided a clear definition of the phases in large-scale projects, as outlined in Table 3. This served as a reference point for annotators to label each window of news articles according to whether it represented a turning point or not.

Due to the diversity of large-scale technology projects, in order to make the dataset more clear and standardized, we categorize the projects in the dataset into three main types: Deep Space Exploration Projects, Large Ground Infrastructure Projects, and Aeronautic and Marine Engineering Projects. Deep Space Exploration (DSE) Projects: These projects are aimed at conducting deep space

scientific research and exploring new technologies. They include celestial exploration and space astronomy research, among others. Large Ground Infrastructure (LGI) Projects: These projects involve the development, maintenance, or improvement of critical large-scale ground infrastructure. This category includes projects related to transportation networks, power systems, nuclear test facilities, network and computing facilities, and ground-based observatories, among others. Aeronautic and Marine Engineering (AME) Projects: These projects are focused on the design, development, and implementation of aeronautic and marine engineering solutions. This category includes projects related to aircraft and ship design, manufacturing, and usage, among others.

The partial statistics of our dataset are presented in Table 4. In total, we collected data from 80 different projects. On average, there was a time span of 53 days between successive news articles within a news stream, indicating that the data within news streams was sparse. To address this issue and to ensure that turning point windows were not missed, we adopted an overlapping window segmentation strategy. For more detailed information about the dataset, please refer to Appendix B.

We allocated a total of 64 projects to the training set, 8 to the development set, and another 8 to the test set. Within these sets, there were 654, 83, and 85 turning point windows, and 3564, 460, and 461 non-turning point windows, respectively.

### 4.2 Baseline

We employed DSVDD(Ruff et al., 2018) as our baseline model and RoBERTa(Liu et al., 2019b) as our strong baseline model.

To validate the effectiveness of each proposed component, we excluded them one by one from our model: (1) w/o InCoder: Excluding the Instance Encoder module. (2) w/o CTE: Excluding the Transformers module with convolution.

**Experiment Settings:** We followed the hyperparameters of RoBERTa-Base (125M parameters) and initialized our model using public pre-trained checkpoints. We set the learning rate to  $8e-4$  and the batch size to 8. For the Baseline model DSVDD, we set the learning rate to  $8e-4$  and the batch size to 64. For the strong Baseline model RoBERTa, we set the learning rate to  $8e-4$  and the batch size to 16.

Table 1: Experimental Results on the constructed dataset.

| Model           | Overall Results |             |             | DSE Projects |             |             | LGI Projects |             |             | AME Projects |             |             |
|-----------------|-----------------|-------------|-------------|--------------|-------------|-------------|--------------|-------------|-------------|--------------|-------------|-------------|
|                 | P               | R           | F1          | P            | R           | F1          | P            | R           | F1          | P            | R           | F1          |
| Bin. Classifier | 0.18            | 0.05        | 0.08        | 0.28         | 0.05        | 0.09        | 0.40         | 0.18        | 0.25        | 0.18         | 0.05        | 0.07        |
| DSVDD           | 0.24            | <b>0.80</b> | 0.36        | 0.34         | 0.79        | 0.48        | 0.22         | <b>0.82</b> | 0.35        | 0.13         | 0.60        | 0.21        |
| RoBERTa         | 0.60            | 0.67        | 0.63        | 0.58         | 0.67        | 0.63        | 0.43         | 0.72        | 0.54        | 0.32         | 0.62        | 0.42        |
| TP-Detector     | <b>0.70</b>     | 0.77        | <b>0.74</b> | <b>0.80</b>  | 0.75        | <b>0.77</b> | <b>0.61</b>  | 0.68        | <b>0.64</b> | <b>0.34</b>  | <b>0.83</b> | <b>0.48</b> |
| w/o InCoder     | 0.67            | 0.75        | 0.71        | 0.60         | <b>0.83</b> | 0.69        | 0.60         | 0.67        | 0.60        | 0.33         | 0.79        | 0.47        |
| w/o CTE         | 0.69            | 0.76        | 0.72        | 0.74         | 0.69        | 0.71        | 0.51         | 0.78        | 0.61        | 0.26         | 0.81        | 0.39        |

### 4.3 Evaluation

For the TP detection task, our goal is to minimize the possibility of overlooking TP windows and accept a certain level of false positives if necessary. Therefore, we aim to achieve a higher recall rate while maintaining the precision of the prediction results. Based on this objective, we observed significant improvements in our model compared to the baseline model DSVDD and the strong Baseline model RoBERTa, which demonstrates the effectiveness of our model in identifying TP windows. For Deep Space Exploration (DSE) projects, our prediction results perform well. There are two main reasons for this: (1) DSE projects are space exploration initiatives managed by organizations like NASA and ESA. These agencies adhere to strict standard procedures and rigorous review processes for their projects. Therefore, these projects generally follow well-defined engineering processes, making the detection of turning points relatively straightforward. (2) There is a relatively large number of DSE projects, and information about them is regularly and comprehensively released by these organizations. Media outlets also tend to focus more on these projects, resulting in a wealth of available information. The abundance of information allows the model to extract more features, which is advantageous for successful turning point detection. For Large Ground Infrastructure (LGI) and Aeronautic and Marine Engineering (AME) projects, our prediction results perform less well. There are several reasons for this: (1) LGI and AME projects are relatively underrepresented in the dataset due to their smaller numbers, which makes it challenging for the model to learn robust patterns specific to these categories. (2) LGI projects generally receive less attention from media organizations compared to DSE projects. Consequently, there is less news coverage for these types of projects, leading to a limited amount of available information. (3)

In AME projects, there are multiple projects with multiple end targets. For instance, in aeronautic engineering projects, the goal is to deliver multiple identical aircraft. Because each aircraft follows the same fixed process from manufacturing to delivery, we consider the first moment of this process as the turning point. However, in such cases, it is challenging for the model to accurately distinguish between these instances.

For the ablation study, we can draw the following conclusions: (1) Instance Encoder: This serves as an encoder for the instances within a bag, with its primary function being to fuse feature representations of multiple news articles within the instances and generate a feature representation for each instance. (2) CTE (Convolutional Transformer Encoder): CTE is a Transformer encoder with convolutional components. Its role is to focus on and enhance the feature representations of both bags and the instances within them.

## 5 Demonstration

We provide our services in a web-based. We deployed the proposed model<sup>4</sup> and provided a demonstration video<sup>5</sup> to illustrate its functionality.

We offer two detection options: Single Project and Multiple Project. In the "Single Project" mode, the system performs TP detection on an individual project. The input entails a JSON file containing the news stream, while the output encompasses TP detection results image and the highest-scoring instance within TP windows. The explanation for a TP window is provided by showcasing the highest-scoring instance. On the other hand, the "Multiple Projects" mode enables simultaneous TP detection across multiple projects. Similar to the single project mode, the input comprises JSON files containing news streams for multiple projects, and

<sup>4</sup><http://43.138.60.114:7080/>

<sup>5</sup><https://youtu.be/FH3av84I-Kg>

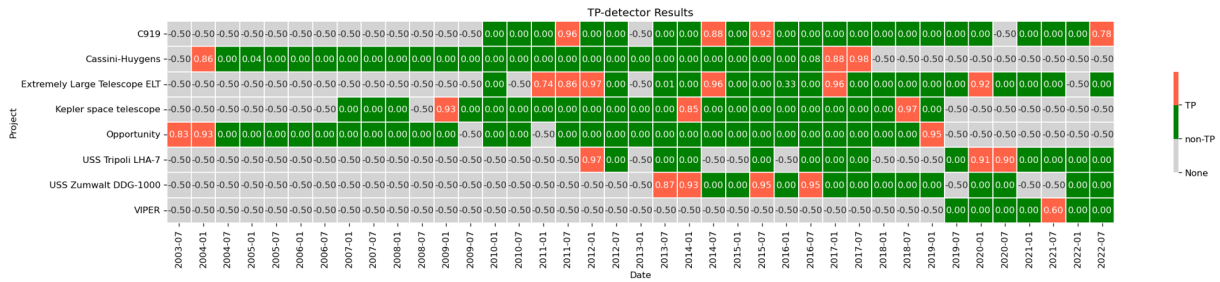


Figure 3: A result example from our demo website.

Table 2: The instance with highest score in the third TP window of the Kepler space telescope project.

| Date            | News content   |
|-----------------|--|
| Oct. 30th, 2018 | NASA's Kepler Space Telescope mission has officially ended. All good things must come to an end, on Earth and even in space. NASA announced on Tuesday that the Kepler mission - which has transformed how we understand planets outside of our solar system - is officially over. According to the space agency, Kepler has run out of fuel in space, ending its 9.5-year planet hunting mission. |
| Oct. 31st, 2018 | NASA retires planet-hunting Kepler space telescope. NASA on Tuesday announced the demise of its elite planet-hunting telescope just a few months shy of its 10th anniversary. The Kepler space telescope that found thousands of planets beyond our solar system and boosted the search for worlds that might support life has run out of fuel.  |

the output is similar to the "Single Project" mode. Figure 3 illustrates the results of Multiple Project detection.

Table 2 presents evidence for the correctness of a detection result. It displays the highest-scored Instance in the third TP window of the prediction results for the Kepler Space Telescope project. This TP occurs between the Operation phase and the Termination phase. Each row in the table represents a news. The first news describes NASA's official retirement of the Kepler space telescope. The second similarly discusses a similar topic. In this instance, the explanation centers around the decision to retire Kepler due to fuel exhaustion, which cannot be replenished. This crucial event has caused the transition of Kepler from an operational state to a terminated state. This crucial event serves as the turning point that we are interested in identifying.

## 6 Conclusion

To investigate phase transitions in engineering processes, we propose the Turning Point Detection

task on large-scale projects. For this task, we introduce a deep multi-instance learning model. This model initially performs feature extraction on input windows using a pre-trained language model. Subsequently, it employs an Instance Encoder to capture instance-level features. Following this, a Convolutional Transformers Encoder is employed to detect phase transition features. Ultimately, a linear head is employed to provide prediction outcomes. We collected a new dataset specifically for this task. Extensive experiments demonstrated that our model outperforms strong baselines. We have deployed our model online to assist in detecting phase transitions in news streams. We hope that our work will contribute to further research in this emerging task and benefit relevant stakeholders. The dataset we have constructed currently only includes publicly available news reports. In the future, we aim to incorporate officially released information into the dataset as an essential supplement to enhance our understanding of the projects with more comprehensiveness.

## Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61976221).

## References

- nsf21107 Research Infrastructure Guide (RIG) (December 2021) | NSF - National Science Foundation — nsf.gov. [https://www.nsf.gov/publications/pub\\_summ.jsp?ods\\_key=nsf21107](https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf21107). [Accessed 07-08-2023].
- Samaneh Aminikhanghahi and Diane J Cook. 2017. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367.
- Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2002. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, 15.

- D Banesh, M Petersen, J Wendelberger, J Ahrens, and B Hamann. 2019. Comparison of piecewise linear change point detection with traditional analytical methods for ocean and climate data. *Environmental Earth Sciences*, 78:1–16.
- W Beitz, G Pahl, and K Grote. 1996. Engineering design: a systematic approach. *Mrs Bulletin*, 71.
- F Lawrence Bennett. 2003. *The management of construction: a project life cycle approach*. Routledge.
- Xinmiao Ding, Bing Li, Weiming Hu, Weihua Xiong, and Zhenchong Wang. 2013. Horror video scene recognition based on multi-view multi-instance learning. In *Computer Vision—ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5–9, 2012, Revised Selected Papers, Part III 11*, pages 599–610. Springer.
- 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*.
- Tom Gilb. 2005. *Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using Planguage*. Elsevier.
- Carlo Grillenzoni. 2012. Evaluation of recursive detection methods for turning points in financial time series. *Australian & New Zealand journal of statistics*, 54(3):325–342.
- Wei He and Yu Wang. 2009. Text representation and classification based on multi-instance learning. In *2009 International Conference on Management Science and Engineering*, pages 34–39. IEEE.
- Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, Sarah Vluymans, Francisco Herrera, Sebastián Ventura, Rafael Bello, et al. 2016. *Multiple instance learning*. Springer.
- Steven R Hirshorn, Linda D Voss, and Linda K Bromley. 2017. Nasa systems engineering handbook. Technical report.
- Bin Li, Yin Li, and Kevin W Eliceiri. 2021. Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14318–14328.
- Shuo Li, Fang Liu, and Licheng Jiao. 2022. Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1395–1403.
- Bo Liu, Yanshan Xiao, and Zhifeng Hao. 2018. A selective multiple instance transfer learning method for text categorization problems. *Knowledge-Based Systems*, 141:178–187.
- Chang Liu, Mark Last, and Armin Shmilovici. 2019a. Identifying turning points in animated cartoons. *Expert Systems with Applications*, 123:246–255.
- 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*.
- Zvi Lotker. 2016. The tale of two clocks. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 768–776.
- Oded Maron and Tomás Lozano-Pérez. 1997. A framework for multiple-instance learning. *Advances in neural information processing systems*, 10.
- James N Martin. 2000. Processes for engineering a system: an overview of the ansi/eia 632 standard and its heritage. *Systems Engineering*, 3(1):1–26.
- Tracy L Osborne. 2022. Nasa space flight program and project management handbook.
- Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2019. Movie plot analysis via turning point identification. *arXiv preprint arXiv:1908.10328*.
- Gwenolé Quéllec, Guy Cazuguel, Béatrice Cochener, and Mathieu Lamard. 2017. Multiple-instance learning for medical image and video analysis. *IEEE reviews in biomedical engineering*, 10:213–234.
- NASA Procedural Requirement. 2018. 7120.8 a, nasa research and technology program and project management requirements”.
- Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR.
- PJ Sudharshan, Caroline Petitjean, Fabio Spanhol, Luiz Eduardo Oliveira, Laurent Heutte, and Paul Honeine. 2019. Multiple instance learning for histopathological breast cancer image classification. *Expert Systems with Applications*, 117:103–111.
- Waqas Sultani, Chen Chen, and Mubarak Shah. 2018. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488.
- Huimin Tang, Peiwu Dong, and Yong Shi. 2019. A new approach of integrating piecewise linear representation and weighted support vector machine for forecasting stock turning points. *Applied soft computing*, 78:685–696.

Yuan Tian, Wenning Hao, Dawei Jin, Gang Chen, and Ao Zou. 2020. A review of latest multi-instance learning. In *Proceedings of the 2020 4th International Conference on Computer Science and Artificial Intelligence*, pages 41–45.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing*, 167:107299.

Yang Yang, Yanlun Tu, Houchao Lei, and Wei Long. 2023. Hamil: Hierarchical aggregation-based multi-instance learning for microscopy image classification. *Pattern Recognition*, 136:109245.

## A Large-scale Project Phases

Table 3: Phase definitions in the engineering process of large-scale projects.

| Phase                   | Description  |
|-------------------------|--|
| Conceptual Studies      | Propose ideas or concepts; Assess the feasibility of ideas or concepts; Establish the requirements and objectives of the task                          |
| Preliminary Design      | Establish baseline tasks; Design architecture; Determine required technologies; Establish a design solution; complete 'implementation' level of design |
| Final Design            | Establish complete, validated detailed design; Complete all design specialty audits; Establish manufacturing processes and controls                    |
| Fabrication&Integration | Prepare production facilities; Manufacture products that meet specifications and acceptance standards; Assemble and integrate systems                  |
| Commissioning           | Validate the system; System test and commissioning   |
| Operation               | Perform mission; Sustain system  |
| Renewal/Pause           | Improve/augment system; Suspend system operations  |
| Termination             | Implement decommission/disposal  |

Table 3 provides the definitions of task phases within the context of large-scale projects. These definitions have been developed by drawing upon the insights and guidelines provided by NASA(Hirshorn et al., 2017; Requirement, 2018; Osborne, 2022) and NSF(nsf) regarding phase definitions in projects. Our approach involved a comprehensive review and synthesis of the definitions from these reputable sources to formulate a clear and concise delineation of the various phases that characterize large-scale projects.

## B Dataset Statistic

Table 4 presents the statistical information for the dataset. There are a total of 80 projects in the

Table 4: Statistics of constructed dataset.

| Project Type             | All  | DSE  | LGI  | AME  |
|--------------------------|------|------|------|------|
| # of Project             | 80   | 46   | 15   | 19   |
| Avg. # of phases         | 3.8  | 4.1  | 2.9  | 3.7  |
| Avg. # of windows        | 66.3 | 62.2 | 60.4 | 81.1 |
| Avg. # of TP windows     | 10.5 | 10.8 | 9.3  | 10.6 |
| Avg. # of non-TP windows | 55.8 | 51.4 | 51.1 | 70.4 |
| Avg. # of news in window | 5.0  | 5.0  | 4.3  | 5.4  |
| Avg. # of news in phase  | 14.0 | 14.3 | 12.3 | 13.9 |
| Avg. phase time span(M)  | 30   | 28   | 35   | 33   |
| Avg. 2 news time span(D) | 53   | 53   | 69   | 47   |

dataset, with 46 Deep Space Exploration Projects, 15 Large Ground Facilities Projects, and 19 Aeronautic and Marine Engineering Projects.

# CLEVA: Chinese Language Models EVALuation Platform

Yanyang Li<sup>1\*</sup>, Jianqiao Zhao<sup>1\*</sup>, Duo Zheng<sup>1</sup>, Zi-Yuan Hu<sup>1</sup>, Zhi Chen<sup>3</sup>, Xiaohui Su<sup>3</sup>,  
Yongfeng Huang<sup>1</sup>, Shijia Huang<sup>1</sup>, Dahua Lin<sup>2,3</sup>, Michael R. Lyu<sup>1</sup>, Liwei Wang<sup>1†</sup>

<sup>1</sup>Department of Computer Science and Engineering, The Chinese University of Hong Kong

<sup>2</sup>Department of Information Engineering, The Chinese University of Hong Kong

<sup>3</sup>Shanghai AI Laboratory

## Abstract

With the continuous emergence of Chinese Large Language Models (LLMs), how to evaluate a model’s capabilities has become an increasingly significant issue. The absence of a comprehensive Chinese benchmark that thoroughly assesses a model’s performance, the unstandardized and incomparable prompting procedure, and the prevalent risk of contamination pose major challenges in the current evaluation of Chinese LLMs. We present CLEVA, a user-friendly platform crafted to holistically evaluate Chinese LLMs. Our platform employs a standardized workflow to assess LLMs’ performance across various dimensions, regularly updating a competitive leaderboard. To alleviate contamination, CLEVA curates a significant proportion of new data and develops a sampling strategy that guarantees a unique subset for each leaderboard round. Empowered by an easy-to-use interface that requires just a few mouse clicks and a model API, users can conduct a thorough evaluation with minimal coding. Large-scale experiments featuring 23 Chinese LLMs have validated CLEVA’s efficacy. Our GitHub repo is <https://github.com/LaVi-Lab/CLEVA>.

## 1 Introduction

Large language models (LLMs) have fundamentally revolutionized natural language processing. Transformer models with more than 100B parameters have exhibited remarkable generalization ability across diverse tasks without the need for fine-tuning. The success of GPT-4 (OpenAI, 2023) and ChatGPT sparked a trend of training Chinese LLMs, with new models launching almost every week (Zeng et al., 2023; Team, 2023; Chenghao Fan and Tian, 2023; Ji et al., 2023; Cui et al., 2023). These rapid developments aggravate the need for Chinese LLM evaluation.

\*Equal contributions.

†Project leader and corresponding author

## Assessing the capacity of LLMs is non-trivial.

Traditional practices that evaluate models on a single task at a time are gradually becoming obsolete, since a single task can hardly characterize a full view of an LLM’s capacity. Instead, to effectively grasp a holistic view of an LLM’s capacity, we need to decompose its capacity into various abilities, evaluate these abilities with numerous corresponding tasks, and measure the competence of each task with multiple metrics. In this sense, HELM (Liang et al., 2022), leads the way in English LLM evaluation, as it conducts an in-depth evaluation of English LLMs on various NLP tasks using seven metrics. In Chinese, previous attempts have shown limitations, either in task selection or the metrics adopted. C-Eval (Huang et al., 2023), M3KE (Liu et al., 2023), CMMLU (Li et al., 2023), GAOKAO-Bench (Zhang et al., 2023), and MMCU (Zeng, 2023) narrow down to knowledge and reasoning abilities, whose datasets are mostly constructed using Chinese exams. By the time of our submission, OpenCompass (Contributors, 2023b), with around 74K Chinese queries out of 300K total, leans on accuracy as its sole metric, overlooking other important aspects in LLM evaluation. FlagEval (Contributors, 2023a) inherits four out of seven metrics from HELM and 22 existing Chinese datasets, having limited coverage on some significant tasks. A comprehensive Chinese benchmark incorporating diverse metrics to holistically evaluate Chinese LLMs is urgently demanded.

## Prompt-based evaluation in Chinese is largely unstandardized.

Previous evaluations, such as HELM (Liang et al., 2022), do not explicitly optimize prompts, though LLMs’ significant sensitivity to the format of prompt has been observed (Webson and Pavlick, 2022; Abdou et al., 2022; Sanh et al., 2022). Moreover, unlike many English benchmarks that have well-developed prompts (§ 3), many Chinese benchmarks are in their early stage and do not enjoy such privileges. Chinese LLMs are eval-

uated using different prompts, making the results incomparable and hence untrustworthy.

Consuming up to trillions of tokens during pre-training, LLMs are prone to train-test contamination (Brown et al., 2020), which significantly threatens the validity of an evaluation. Previous work (OpenAI, 2023; Liang et al., 2022) approaches this issue more from a consequentialist perspective: They examine the contamination risk, by methods like long n-gram overlap, only after the evaluation has been done. These post-evaluation analyses, though responsibly examining if train-test contamination happens, cannot alleviate the risk of contamination in the first place. A proactive method to mitigate the contamination risk is of great importance.

We present CLEVA, Chinese Language models EVAluation platform that tackles the aforementioned problems with the following features:

- **A comprehensive Chinese benchmark.** Inspired by HELM (Liang et al., 2022), CLEVA organizes the evaluation tasks into two parts: *ability evaluation*, which gauges specific LLM skills and *application assessment*, which tests how well LLMs apply their skills to real-world applications (§ 4.1). Most of the well-accepted Chinese datasets relevant to our ability evaluation or application assessment are organized, standardized, and then adopted by our platform. More importantly, we design new Chinese-specific tasks, e.g., Pinyin transliteration and intent understanding, and collect a substantial amount of new data, accounting for 33.98% of our total data. As for the metrics (§ 4.1), we incorporate metrics for diversity and privacy into our system in addition to the seven in HELM. With 370K (over 9 million queries after augmentation) test instances from 84 datasets and 9 metrics, CLEVA, so far, stands as the most extensive Chinese evaluation dataset and possesses the most dimensions, facilitating a holistic evaluation of Chinese LLMs.
- **Standardized prompt-based evaluation methodology.** CLEVA takes full control of key aspects of LLM evaluation, with data and prompts being the most important among them. All data are jointly prepared with unified preprocessing steps, ensuring a level playing field for all LLMs. Meanwhile, CLEVA provides a set of prompts, instead of just

one prompt as in prior work, for each task for prompting-based inference (Brown et al., 2020). This prompt design ensures comparable evaluation results by standardizing the prompts used for testing, while also encouraging further analysis of LLMs’ sensitivity to different prompts (Zhu et al., 2023).

- **An up-to-date and trustworthy leaderboard.** CLEVA advocates a proactive method for securing trustworthy evaluation results. By collecting extensive new data, CLEVA suppresses the leakage of testing data prior to the evaluation. Moreover, we frequently organize new evaluation rounds, sampling a unique test set from 9 million augmented instances. This strategy further mitigates the risk of train-test contamination, improving the trustworthiness and timeliness of the leaderboard.

CLEVA is thoroughly validated by benchmarking 23 Chinese LLMs on our large-scale test sets (§ 6). The corresponding leaderboard and all other user-friendly features will be continuously maintained and improved to accommodate new models and evaluation methods.

## 2 Related Work

LLM evaluation is a rapidly developing field in recent years to delineate the boundary of LLM’s capability. In English, various systematic evaluation benchmarks have been proposed. For example, BIG-Bench (bench authors, 2023) is the largest collection that covers more than 200 tasks. HELM (Liang et al., 2022) organizes tasks into core scenarios, which focus on use cases, and targeted evaluation, which aims to better understand models. HELM also presents a multi-metric measurement that enables analysis of tradeoffs for each scenario. Recently, AGIEval (Zhong et al., 2023) is proposed to evaluate LLMs using challenging human exams. PromptBench (Zhu et al., 2023), on the other hand, measures the robustness of LLMs to prompts via adversarial attacks. MT-Bench (Zheng et al., 2023) collects multi-turn questions and presents the Chatbot Arena platform that treats GPT-4 (OpenAI, 2023) as the judge.

While CLEVA shares the same fundamental motivation with HELM (Liang et al., 2022), to holistically evaluate language learning models in their original languages, CLEVA is far from a mere Chinese replica of HELM. Building on the foundation of HELM’s taxonomy, CLEVA introduces a range

of tasks, with particular emphasis on those unique to Chinese, to better assess the capabilities of Chinese LLMs. It offers a new perspective on prompts, providing abundant prompt templates to standardize evaluation and encourage in-depth exploration of models’ sensitivity. In terms of metrics, CLEVA expands into new areas of diversity and privacy for a more comprehensive evaluation. Finally, CLEVA proactively mitigates train-test contamination by collecting a significant amount of new data, creating unique test sets by sampling, and regularly updating the leaderboard. All of these evaluation designs are neatly packaged in a user-friendly platform to facilitate community usage.

There is also a lot of progress in evaluating Chinese LLMs (Huang et al., 2023; Liu et al., 2023; Li et al., 2023; Zhang et al., 2023; Zeng, 2023). OpenCompass (Contributors, 2023b) and FlagEval (Contributors, 2023a) are two important attempts to evaluate Chinese LLMs. OpenCompass pools 53 public datasets and uses standard accuracy-like metrics as the only measurement for each dataset. FlagEval, with a smaller number of datasets and metrics, still needs further expansion to achieve sufficient coverage. Compared to previous efforts, CLEVA offers Chinese data from 84 datasets, including 33.98% original queries, while employing the broadest range of metrics to promote holistic evaluation. CLEVA standardizes prompts (§ 4) and mitigates data contamination issues, pioneering new paths for LLM evaluation in general.

### 3 Preliminaries

To measure the model performance on a task, a relevant *test set* is constituted from a collection of *instances*. A test instance will contain multiple *input fields* (string typically) and a list of *references*.

We then adopt a *prompt template*, which essentially describes how to assemble the model input, a.k.a, *prompt*, from input fields (Bach et al., 2022). For example, a Chinese paraphrase identification prompt template (and its translation) is:

**Chinese Example:**

“{sentence1}”和“{sentence2}”这两个问题是在问同一件事情吗?

**English Translation:**

Are the questions “{sentence1}” and “{sentence2}” asking the same thing?

where {sentence1} and {sentence1} are two input fields that will be replaced by the two candidate questions in the test instance. The prompt will

be fed into a black-box LLM to predict an output string together with its probability.

Finally, all model predictions and the corresponding test instances will be passed into a *metric* to obtain a numerical value that indicates how well the model performs. Following HELM (Liang et al., 2022), a *metric* in this paper is an umbrella for a dimension of measures that share similar purposes. For example, the *accuracy* metric corresponds to BLEU for translation and pass@k for code synthesis. We employ nine metrics, foregrounding metrics beyond accuracy and ensuring a holistic evaluation.

## 4 System Design

CLEVA aims to deliver the following two key assets to users who try to evaluate their own LLMs:

- A comprehensive and thorough **assessment report** that informs users of the strength and limitations of their models.
- A trustworthy **leaderboard** reflecting the latest advancement of LLMs.

We will discuss our taxonomy that ensures comprehensive evaluations, and challenges like train-test contamination in leaderboard maintenance.

### 4.1 Evaluation Taxonomy

Inspired by HELM (Liang et al., 2022), we present a **Tasks**×**Prompts**×**Metrics** evaluation taxonomy for users to evaluate their models. Our evaluation taxonomy carefully designs a Chinese benchmark targeting various LLM abilities, employs a set of diverse prompt templates for each task to characterize the model performance variance, and adopts multiple metrics to comprehensively assess LLMs. **Tasks.** As shown in Figure 1, our Chinese LLM evaluation benchmark consists of two parts: *ability evaluation* and *application assessment*. Each task in ability evaluation focuses on one special skill of LLMs, while application assessment involves real-world NLP tasks that require LLMs to solve practical use cases with their skill sets. Ability evaluation assesses LLM ability from five aspects:

- **Language** measures how well LLMs understand Chinese. In addition to three conventional tasks, we incorporate two tasks specific to Chinese: *Pinyin transliteration* and *classical Chinese understanding*.
- **Knowledge** focuses on assessing the capacity of knowledge acquired by LLMs. We further segment our evaluation into *subject*



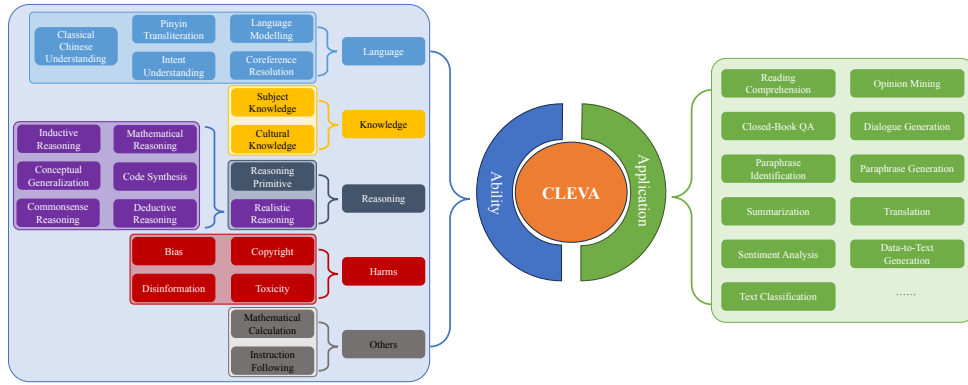


Figure 1: CLEVA benchmark.

*knowledge* and *cultural knowledge* (mainly Chinese culture) based on the source of knowledge. This fine-grained design allows users to closely analyze the model performance across different knowledge categories.

- **Reasoning** evaluates LLMs’ reasoning ability in two settings: *reasoning primitives*, which is independent of language and knowledge background, and *realistic reasoning* that requires reasoning with domain knowledge on practical scenarios. On top of HELM, we additionally include *commonsense reasoning*, *inductive reasoning*, *conceptual generalization*, and *deductive reasoning*.
- **Harms** evaluates the potential risk of LLMs in *copyright*, *disinformation*, *bias*, and *toxicity*.
- **Others** is newly introduced to include crucial yet uncategorized tasks like *mathematical calculation* and *instruction following*.

For application assessment, CLEVA features 11 real-world NLP tasks. In addition to the core scenarios of HELM, we newly include *opinion mining*, *dialogue generation*, *paraphrase generation*, *translation*, *paraphrase identification*, and *data-to-text generation*. A detailed description of each task is documented in Appendix B.

We instantiate the aforementioned tasks in two ways: by directly adopting related public Chinese datasets and by collecting new data. For well-studied tasks, widely-recognized datasets are the best options for forming our benchmark. However, many important tasks, such as *reasoning primitive*, *Pinyin transliteration*, and *disinformation*, lack corresponding Chinese datasets, making the evaluation even more challenging. On these occasions, we either synthesize using sophisticated rule-based scripts (e.g., reasoning primitive) or enlist professional human annotators to construct new test sets (See Appendix C for annotation details). In total,

the 31 tasks include 370K test instances from 84 datasets (9M queries in total after applying multiple prompt templates and data augmentation), 33.98% of which are newly collected.

**Prompts.** Ideally, an LLM should be a general interface, capable of understanding prompts with the same semantics, regardless of variations in surface forms. However, LLMs’ notorious sensitivity to prompt templates hinders accurate evaluation (Webson and Pavlick, 2022; Abdou et al., 2022), leading to results that are sometimes incomparable. To better understand an LLM’s sensitivity to plausible human instructions, multiple prompt templates are needed, rather than a single template as in previous work (Contributors, 2023a,b; Liang et al., 2022).

In this work, we manually annotate an average of 3.95 prompt templates for each test set and support all major prompting formats. CLEVA calculates the performance statistics across the entire set of prompts. These statistics do more than just examine the robustness to prompt templates, as reflected by the standard deviation; they also help estimate the upper and lower bounds of an LLM’s performance on a specific test set, as indicated by the minimum and maximum values. Users can benefit from these statistics to select models and to make informed trade-offs between performance and investment in prompt engineering. More discussions on prompt templates we provided are in Appendix F.

**Metrics.** We adopt the 7 metrics from HELM for a holistic evaluation, and, to address recent interest in chatbots and safety concerns, we add two new dimensions: *diversity* and *privacy*.

- **Accuracy.** Accuracy refers to the standard metrics to measure model performance on different tasks, e.g., F1 score for question answering and ROUGE score for summarization.
- **Calibration and uncertainty.** Calibration represents the gap between the model confi-

dence and its actual error rate and is measured mainly by expected calibration error (ECE, (Naeini et al., 2015)).

- **Robustness.** Robustness is the worst-case performance of a model across transformations of test instances. We focus on semantics-preserving perturbations as there are many well-studied data augmentation tools.
- **Fairness.** Similar to robustness, fairness employs perturbations related to social groups to test the disparate treatment and disparate impact of LLMs.
- **Bias and stereotypes.** We quantify bias as the disproportionate representation of different social groups. This is gauged through the rates at which these groups are mentioned during model generation. Additionally, we interpret stereotypes as uneven associations between these social groups and certain stereotyped terms, such as occupational roles.
- **Toxicity.** Following HELM (Liang et al., 2022), toxicity is a general term that covers hate speech, abusive language, etc.
- **Efficiency.** Efficiency is a rather broad concept that has many subtleties. It could refer to training or inference efficiency and is measured by energy, carbon, and wall-clock time. As most information could be confidential, we focus only on the inference wall-clock time.
- **Diversity.** Given the popularity of LLM-based chatbots, we incorporate the conventional diversity metric in dialogue systems that evaluates the response surface form diversity (Li et al., 2016). Here, we employ the diversity metrics from Miller et al. (2017).
- **Privacy.** In the real-world deployment of LLMs, detecting private information in the generated text, such as Personally Identifiable Information (PII), is a challenging yet important question. We report the portion of PII in the whole test set to make the privacy evaluation generalizable. CLEVA adopts some established tools to smoothly detect PII, and we are working on accommodating more aspects of private content in the near future.

Detailed metric lists are provided in Appendix D.

## 4.2 Leaderboard & Data Contamination

Ensuring fairness, objectivity, and authority is central to maintaining a trustworthy leaderboard. Previous work (Brown et al., 2020) has reported **train-**

**test contamination**, a situation where the test set is included in the training data, leading to unreliable evaluations. Many existing benchmarks, e.g., Huang et al. (2023), conceal the test set labels to avoid data contamination. Given the small scale of their test sets and the large-scale training corpora used by modern LLMs, the risk of unintentional train-test contamination remains high. Sun et al. (2023a) address this problem by making the official test set private and requiring users to submit models’ weights for evaluation. However, this arrangement is unpopular because numerous cutting-edge models consider their weights highly confidential.

We advocate “mutual confidentiality” in LLM evaluation: Users need not expose their model details, and the platform should minimize the risk of disclosing its test set. Instead of model weights, CLEVA only requires API access. We proactively achieve the other half of mutual confidentiality by continuously collecting new data and frequently organizing leaderboard rounds with unique test sets sampling from our full-scale 9 million augmented instances. These strategies not only improve evaluation efficiency but also alleviate train-test contamination from data and temporal perspectives.

To make sure that the sampled subset delivers accurate results, our sampling strategy is not just random sampling: It estimates an acceptable approximation error threshold (i.e., within this threshold, the evaluation results on the sampled set have at least a 70% chance to correctly rank any model pairs), then adjusts the sampling rate for each task according to this threshold, reducing the risk of over-/under-estimating the model performance.

## 5 Usage Example

Upon authentication, users are immediately presented with an interactive summary of our evaluation results of 23 LLMs. Users can select from these models, freely exploring the evaluation results from all 9 metrics and 31 tasks.

CLEVA simplifies the evaluation process of new models with minimal coding required. If a user has a model to evaluate, the user only needs a few minutes to finish these three steps: entering the model’s API, selecting relevant tasks from 31 choices, and picking desired metrics from 9 options. CLEVA will autonomously call the user’s model, extract the corresponding responses, and compute the final metrics. Detailed descriptions and screenshots of CLEVA are listed in Appendix A.

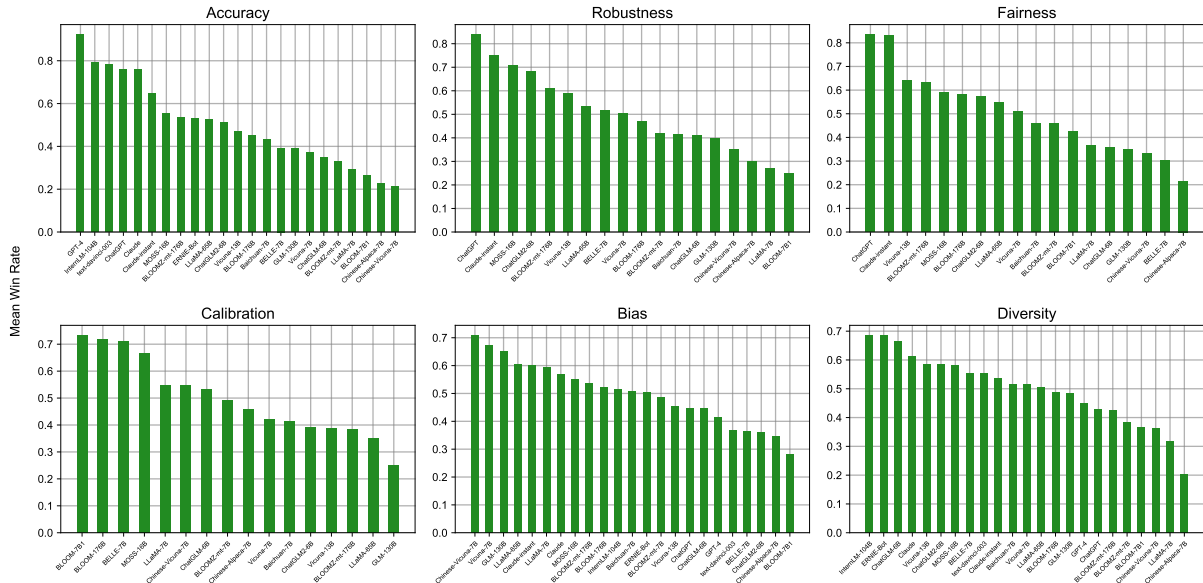


Figure 2: The mean win rate of 23 models in 31 tasks. The mean win rate is the probability of a model outperforming a random different model on a random task. We exclude toxicity, privacy, and efficiency metrics as all models excel in the former two, and the latter is often paired with other metrics to deliver meaningful comparisons. Since robustness and fairness involve expensive data augmentation, we only evaluate ChatGPT and Claude-instant.

## 6 Evaluation

**Setup.** We sample 6.43% of our data to test 23 models that support Chinese (See Appendix E). As for the cost, for example, it takes roughly 1600 GPU hours (NVIDIA A100 80G) to evaluate BLOOMZ-176B-mt (Muennighoff et al., 2023).

**Results & Analysis.** Figure 2 ranks all models by their mean win rates under different metrics.

- **Accuracy.** It can be seen that GPT-4 (OpenAI, 2023) has the highest winning rate, followed by other limited-accessed models. This result shows a considerable margin between the performance of open-source models and limited-accessed models. Recent small instruction-following models are better than large LLMs without instruction-tuning, and are even better than some early large instruction-following models, indicating the necessity of effective instruction tuning.
- **Robustness.** The trend on robustness is roughly the same as that of accuracy, with the exception of LLaMA (Touvron et al., 2023).
- **Fairness.** Most of the model rankings have changed. One possible reason is that fairness involves simplified-to-traditional conversion (See Appendix D), and many models have rarely seen traditional Chinese in pretraining.
- **Calibration.** We report ECE-10 (Kumar et al., 2019) following HELM. We find that models with more parameters tend to have higher

ECE. For example, GLM-130B (Zeng et al., 2023) and LLaMA-65B rank at the bottom. For BLOOMZ-mt-7B vs BLOOMZ-mt-176B and BLOOM-7B1 vs BLOOM-176B (Scao et al., 2022), the smaller one wins.

- **Bias.** We focus on gender bias for comparison. GPT-4 and other models, which rank top by other metrics, are at the bottom, while most of the open-source models have low bias. This is because open-source models usually output shorter, resulting in a lower risk of bias.
- **Diversity.** We choose inter-distinct to compare different models. Open-source models generate more diverse and innovative expression than limited-accessed ones, probably due to their fewer safety concerns.

More detailed results and analysis are in Appendix G.

## 7 Conclusion

We present CLEVA, a Chinese LLM evaluation platform. With the largest scale of Chinese instances and broadest metrics, CLEVA provides a comprehensive benchmark to holistically evaluate Chinese LLMs. CLEVA standardizes key components, such as prompt templates, to make evaluation comparable. It also proactively mitigates the contamination issue by collecting large-scale new data, sampling for unique test sets, and regularly updating the leaderboard.

## Limitations

Without further information needed from users, we can only use the inference walk-clock time as the metric, which may have a larger variance when the network is unstable. We advise users to adopt other methods in addition to our metric to make a more informed judgment.

In addition, how to evaluate privacy is still a challenging problem. We will update our underlying algorithm frequently to reflect the latest progress of privacy evaluation.

## Ethics Statement

We consider the ethics issue in two folds, responsible data collection and usage. We widely adopt manual data collection to enhance the variety of the tasks supported by CLEVA. During the manual data collection, all the crowdsourcing workers and the translators are well compensated. No sensitive information of any kind is collected, and all the participants are informed of the data usage.

CLEVA involves tasks that evaluate LLMs' performance on harm. Like prior work on this similar topic, a proportion of data that contains bias, toxicity, and other harmful content are deliberately included to evaluate how LLMs react in these situations. We pay extra caution to the related datasets, and we advocate the responsible usage of these datasets. These datasets should only be used for LLM evaluation. Our sampling mechanism also reduces the unwanted leakage of the data.

## Acknowledgements

This work was supported by National Key R&D Program of China (Project No. 2022ZD0161200, 2022ZD0161201). It was also partially funded by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK. Liwei Wang is a Principal Investigator of CPII under the InnoHK. This work was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund).

## References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. [Word order does matter and shuffled language models know it](#). 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 6907–6919. Association for Computational Linguistics.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. [A general language assistant as a laboratory for alignment](#). *CoRR*, abs/2112.00861.

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](#).

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 B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022a. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#). *CoRR*, abs/2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. [Constitutional AI: harmfulness from AI feedback](#). *CoRR*, abs/2212.08073.

BIG bench authors. 2023. [Beyond the imitation game: Quantifying and extrapolating the capabilities of language models](#). *Transactions on Machine Learning Research*.

- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 491–500. ACM.
- 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*.
- Ben Buchanan, Andrew Lohn, Micah Musser, and Kateřina Sedova. 2021. [Truth, lies, and automation: How language models could change disinformation](#).
- Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. [Extracting training data from large language models](#). In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2633–2650. USENIX Association.
- Wanxiang Che, Yunlong Feng, Libo Qin, and Ting Liu. 2021. [N-LTP: An open-source neural language technology platform for Chinese](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 42–49, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *CoRR*, abs/2107.03374.
- Zhenyi Lu Chenghao Fan and Jie Tian. 2023. [Chinese-vicuna: A chinese instruction-following llama-based model](#).
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%\\* chatgpt quality](#).
- FlagEval Contributors. 2023a. [Flageval](#). <https://github.com/FlagOpen/FlagEval>.
- OpenCompass Contributors. 2023b. [Opencompass: A universal evaluation platform for foundation models](#). <https://github.com/InternLM/OpenCompass>.
- Yiming Cui, Ziqing Yang, and Xin Yao. 2023. [Efficient and effective text encoding for chinese llama and alpaca](#). *CoRR*, abs/2304.08177.
- Jiawen Deng, Jingyan Zhou, Hao Sun, Chujie Zheng, Fei Mi, Helen Meng, and Minlie Huang. 2022. [COLD: A benchmark for chinese offensive language detection](#). 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 11580–11599. Association for Computational Linguistics.
- Kaustubh D. Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Srivastava, Samson Tan, Tongshuang Wu, Jascha Sohl-Dickstein, Jinho D. Choi, Eduard Hovy, Ondrej Dusek, Sebastian Ruder, Sajant Anand, Naganender Aneja, Rabin Banjade, Lisa Barthe, Hanna Behnke, Ian Berlot-Attwell, Connor Boyle, Caroline Brun, Marco Antonio Sobrevilla Cabezudo, Samuel Cahyawijaya, Emile Chapuis, Wanxiang Che, Mukund Choudhary, Christian Claus, Pierre Colombo, Filip Cornell, Gautier Dagan, Mayukh Das, Tanay Dixit, Thomas Dopierre, Paul-Alexis Dray, Suchitra Dubey, Tatiana Ekeinhor, Marco Di Giovanni, Rishabh Gupta, Rishabh Gupta, Louanes Hamla, Sang Han, Fabrice Harel-Canada, Antoine Honore, Ishan Jindal, Przemyslaw K. Joniak, Denis Kleyko, Venelin Kovatchev, Kalpesh Krishna, Ashutosh Kumar, Stefan Langer, Seungjae Ryan Lee, Corey James Levinson, Hualou Liang, Kaizhao Liang, Zhexiong Liu, Andrey Lukyanenko, Vukosi Marivate, Gerard de Melo, Simon Meoni, Maxime Meyer, Afnan Mir, Nafise Sadat Moosavi, Niklas Muennighoff, Timothy Sum Hon Mun, Kenton Murray, Marcin Namysl, Maria Obedkova, Priti Oli, Nivranshu Pasricha, Jan Pfister, Richard Plant, Vinay Prabhu, Vasile Pais, Libo Qin, Shahab Raji, Pawan Kumar Rajpoot, Vikas Raunak, Roy Rinberg, Nicolas Roberts, Juan Diego Rodriguez, Claude Roux, Vasconcellos P. H. S., Ananya B. Sai, Robin M. Schmidt, Thomas Scialom, Tshephiso Sefara, Saqib N. Shamsi, Xudong Shen, Haoyue Shi, Yiwen Shi, Anna Shvets, Nick Siegel, Damien Sileo, Jamie Simon, Chandan Singh, Roman Sitelew, Priyank Soni, Taylor Sorensen, William

- Soto, Aman Srivastava, KV Aditya Srivatsa, Tony Sun, Mukund Varma T, A Tabassum, Fiona Anting Tan, Ryan Teehan, Mo Tiwari, Marie Tolkiehn, Athena Wang, Zijian Wang, Gloria Wang, Zijie J. Wang, Fuxuan Wei, Bryan Wilie, Genta Indra Winata, Xinyi Wu, Witold Wydmański, Tianbao Xie, Usama Yaseen, M. Yee, Jing Zhang, and Yue Zhang. 2021. [NI-augmenter: A framework for task-sensitive natural language augmentation](#).
- 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.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. [GLM: general language model pretraining with autoregressive blank infilling](#). 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 320–335. Association for Computational Linguistics.
- Nan Duan. 2018. [Overview of the NLPCC 2018 shared task: Open domain QA](#). In *Natural Language Processing and Chinese Computing - 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26-30, 2018, Proceedings, Part II*, volume 11109 of *Lecture Notes in Computer Science*, pages 452–456. Springer.
- Ran El-Yaniv and Yair Wiener. 2010. [On the foundations of noise-free selective classification](#). *J. Mach. Learn. Res.*, 11:1605–1641.
- Hao Fu, Yao; Peng and Tushar Khot. 2022. [How does gpt obtain its ability? tracing emergent abilities of language models to their sources](#). *Yao Fu's Notion*.
- 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. 2021a. [The pile: An 800gb dataset of diverse text for language modeling](#). *CoRR*, abs/2101.00027.
- 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.
- Prakhar Gupta, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. [Dialfact: A benchmark for fact-checking in dialogue](#). 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 3785–3801. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence S. Moss. 2020. [OCNLI: original chinese natural language inference](#). *CoRR*, abs/2010.05444.
- Xuming Hu, Zhijiang Guo, GuanYu Wu, Aiwei Liu, Lijie Wen, and Philip Yu. 2022. [CHEF: A pilot Chinese dataset for evidence-based fact-checking](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3362–3376, Seattle, United States. Association for Computational Linguistics.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. [C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models](#). *CoRR*, abs/2305.08322.
- Yunjie Ji, Yong Deng, Yan Gong, Yiping Peng, Qiang Niu, Lei Zhang, Baochang Ma, and Xiangang Li. 2023. [Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases](#). *CoRR*, abs/2303.14742.
- Tom Kocmi, Rachel Bawden, Ondrej 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 Popovic. 2022. [Findings of the 2022 conference on machine translation \(WMT22\)](#). In *Proceedings of the Seventh Conference on Machine Translation, WMT 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 7-8, 2022*, pages 1–45. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *NeurIPS*.
- Ananya Kumar, Percy Liang, and Tengyu Ma. 2019. [Verified uncertainty calibration](#). 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 3787–3798.
- Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Prakash Gupta, Donald Metzler, and Lucy Vasserman. 2022. [A new generation of perspective API: efficient multilingual character-level transformers](#). In

- KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 3197–3207. ACM.
- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. [CMMLU: measuring massive multitask language understanding in chinese](#). *CoRR*, abs/2306.09212.
- Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. [A diversity-promoting objective function for neural conversation models](#). In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.
- Wenhao Li, Fanchao Qi, Maosong Sun, Xiaoyuan Yi, and Jiarui Zhang. 2021. [CCPM: A chinese classical poetry matching dataset](#). *CoRR*, abs/2106.01979.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yuksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. [Holistic evaluation of language models](#). *CoRR*, abs/2211.09110.
- 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.
- Haitao Lin, Liqun Ma, Junnan Zhu, Lu Xiang, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2021. [CSDS: A fine-grained Chinese dataset for customer service dialogue summarization](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4436–4451, 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 *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 3214–3252. Association for Computational Linguistics.
- Chuang Liu, Renren Jin, Yuqi Ren, Linhao Yu, Tianyu Dong, Xiaohan Peng, Shuting Zhang, Jianxiang Peng, Peiyi Zhang, Qingqing Lyu, Xiaowen Su, Qun Liu, and Deyi Xiong. 2023. [M3KE: A massive multi-level multi-subject knowledge evaluation benchmark for chinese large language models](#). *CoRR*, abs/2305.10263.
- Kang Liu, Liheng Xu, and Jun Zhao. 2012. [Opinion target extraction using word-based translation model](#). In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1346–1356. ACL.
- Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. [A survey of deep learning for mathematical reasoning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14605–14631. Association for Computational Linguistics.
- Ian R. McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, Andrew Gritsevskiy, Daniel Wurgaft, Derik Kauffman, Gabriel Recchia, Jiacheng Liu, Joe Cavanagh, Max Weiss, Sicong Huang, The Floating Droid, Tom Tseng, Tomasz Korbak, Xudong Shen, Yuhui Zhang, Zhengping Zhou, Najoung Kim, Samuel R. Bowman, and Ethan Perez. 2023. [Inverse scaling: When bigger isn't better](#). *CoRR*, abs/2306.09479.
- Alexander H. Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. [Parlai: A dialog research software platform](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017 - System Demonstrations*, pages 79–84. Association for Computational Linguistics.
- Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M. Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailley Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. [Crosslingual generalization through multitask finetuning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 15991–16111. Association for Computational Linguistics.
- Mahdi Pakdaman Naeni, Gregory F. Cooper, and Milos Hauskrecht. 2015. [Obtaining well calibrated probabilities using bayesian binning](#). In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2901–2907. AAAI Press.

- Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. [Codegen: An open large language model for code with multi-turn program synthesis](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- 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*.
- 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.
- Roma Patel and Ellie Pavlick. 2022. [Mapping language models to grounded conceptual spaces](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. [True few-shot learning with language models](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 11054–11070.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. [Language models as knowledge bases?](#) In *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 2463–2473. 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, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics.
- Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. [Data-to-text generation with content selection and planning](#). In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6908–6915. AAAI Press.
- 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*. OpenReview.net.
- 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.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). *CoRR*, abs/2302.04761.
- Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. [Long and diverse text generation with planning-based hierarchical variational model](#). 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 3255–3266. Association for Computational Linguistics.
- Shane Storcks, Qiaozhi Gao, and Joyce Y. Chai. 2019. [Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches](#). *CoRR*, abs/1904.01172.
- Hao Sun, Zhexin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023a. [Safety assessment of chinese large language models](#). *CoRR*, abs/2304.10436.



- Hong Sun and Ming Zhou. 2012. **Joint learning of a dual SMT system for paraphrase generation**. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 38–42. The Association for Computer Linguistics.
- Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. **Probing prior knowledge needed in challenging chinese machine reading comprehension**. *CoRR*, abs/1904.09679.
- Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, and Xipeng Qiu. 2023b. **Moss: Training conversational language models from synthetic data**.
- InternLM Team. 2023. **Internlm: A multilingual language model with progressively enhanced capabilities**. <https://github.com/InternLM/InternLM>.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. **FEVER: a large-scale dataset for fact extraction and verification**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. **Llama: Open and efficient foundation language models**. *CoRR*, abs/2302.13971.
- Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. **Can generative pre-trained language models serve as knowledge bases for closed-book qa?** 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 3241–3251. Association for Computational Linguistics.
- Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. **Deep neural solver for math word problems**. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. **Self-instruct: Aligning language models with self-generated instructions**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.
- Albert Webson and Ellie Pavlick. 2022. **Do prompt-based models really understand the meaning of their prompts?** In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2300–2344. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. **Finetuned language models are zero-shot learners**. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. **Emergent abilities of large language models**. *Trans. Mach. Learn. Res.*, 2022.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022c. **Chain-of-thought prompting elicits reasoning in large language models**. In *NeurIPS*.
- 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.
- Liang Xu, Xiaojing Lu, Chenyang Yuan, Xuanwei Zhang, Hu Yuan, Huilin Xu, Guoao Wei, Xiang Pan, and Hai Hu. 2021. **Fewclue: A chinese few-shot learning evaluation benchmark**. *CoRR*, abs/2107.07498.
- Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. **GLM-130B: an open bilingual pre-trained model**. In *The Eleventh International Conference on Learning*

- Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Hui Zeng. 2023. [Measuring massive multitask chinese understanding](#). *CoRR*, abs/2304.12986.
- Bowei Zhang, Weiwei Sun, Xiaojun Wan, and Zongming Guo. 2019. [PKU paraphrase bank: A sentence-level paraphrase corpus for chinese](#). In *Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9-14, 2019, Proceedings, Part I*, volume 11838 of *Lecture Notes in Computer Science*, pages 814–826. Springer.
- Lei Zhang and Bing Liu. 2017. [Sentiment analysis and opinion mining](#). In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1152–1161. Springer.
- Sheng Zhang, Xin Zhang, Hui Wang, Lixiang Guo, and Shanshan Liu. 2018. [Multi-scale attentive interaction networks for chinese medical question answer selection](#). *IEEE Access*, 6:74061–74071.
- Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. [Evaluating the performance of large language models on GAOKAO benchmark](#). *CoRR*, abs/2305.12474.
- Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. [ChID: A large-scale Chinese IDiom dataset for cloze test](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 778–787, Florence, Italy. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). *CoRR*, abs/2306.05685.
- Wanjuan Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. [Agieval: A human-centric benchmark for evaluating foundation models](#). *CoRR*, abs/2304.06364.
- Jingyan Zhou, Jiawen Deng, Fei Mi, Yitong Li, Yasheng Wang, Minlie Huang, Xin Jiang, Qun Liu, and Helen Meng. 2022. [Towards identifying social bias in dialog systems: Frame, datasets, and benchmarks](#). *CoRR*, abs/2202.08011.
- Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, and Xing Xie. 2023. [Promptbench: Towards evaluating the robustness of large language models on adversarial prompts](#). *CoRR*, abs/2306.04528.
- Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020. [Crosswoz: A large-scale chinese cross-domain task-oriented dialogue dataset](#). *Trans. Assoc. Comput. Linguistics*, 8:281–295.
- Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. [Universal and transferable adversarial attacks on aligned language models](#).

## A Platform Usage

To fully utilize our CLEVA to evaluate a large language model, users can take advantage of our user-friendly web application. As shown by Figure 3(a), users will first see our latest leaderboard results with an interactive interface. Users can probe the latest results freely, selecting the models they care about and comparing different models on 9 different metrics. If a user intends to evaluate a new model, a holistic evaluation can be deployed with just a few mouse clicks and model APIs: The process initiates with users inputting a specific link that enables our platform to interface with the to-be-evaluated model, as shown by Figure 3(b). Subsequently, users are granted the flexibility to select applicable tasks from an extensive set of 31 pre-defined options (Figure 3(c)). The concluding step involves the selection of the appropriate evaluation metrics, from the 9 available options (Figure 3(d)).

## B Benchmark

In this section, we provide a detailed description along with an example for each task involved in our benchmark. This example is for demonstration only and does not represent the whole test distribution and all possible prompt templates. We also accompany the English translation after each Chinese example. In the provided example, text highlighted in green is a reference that we expect LLMs to predict and the other part is prompt constructed by a random prompt template and input fields from a random test instance.

### B.1 Ability Evaluation

#### B.1.1 Language

**Language Modelling.** This task asks the LLM to score the probability of the input text. We use bits per bytes (Gao et al., 2021a) as the metric that allows us to make comparisons with different tokenizers. Data are sampled from CLUECorpus2020 (Xu et al., 2020).

**Coreference Resolution.** Coreference resolution is a traditional NLP task. We sample data from CLUEWSC (Xu et al., 2021), where the model must answer whether a given pronoun refers to a given entity (the Winograd Schema Challenge). We use accuracy as the metric for this problem. A coreference resolution example is shown below:

**Chinese Example:**

蒋盈波原来所在的教研室有位副教授去德国参加一个学术活动，活动中结识了一位华裔德籍的同行，那同行在自己

家中招待了他一次，言谈之间，双方忽然都感到巧事真多，而世界真小

在这里，“他”的意思是“同行”。是或否？

**English Translation:**

An associate professor from the research office where Jiang Yingbo used to work went to Germany to attend an academic event. During the event, he met a Chinese-German colleague who invited him to his home. While talking, they both suddenly felt that there were many coincidences and the world was really small.

Here, does “him” refer to “colleague”? Yes or No?

**Pinyin Transliteration.** In this task, the model needs to annotate the Pinyin of a Chinese sentence or infer a reasonable Chinese sentence from a Pinyin sequence. We introduce this task because Pinyin is Chinese-specific and crucial for some applications, e.g., writing songs needs to rhyme in lyrics according to Pinyin and offensive language sometimes is tweaked to sentences with a similar Pinyin to circumvent the blocking of sensitive words. Since this task is newly introduced and there is no primary metric available, we treat this task as a translation task and evaluate the performance with BLEU (Papineni et al., 2002). A Chinese-to-Pinyin transliteration example is shown below:

**Chinese Example:**

将以下句子在汉字和汉语拼音之间进行转译。

汉字：因此，依靠科技进步，强化科学管理已成为实现油田稳产的当务之急

拼音： yīn cǐ, yī kào kē jì jìn bù, qiáng huà kē xué guǎn lǐ

yǐ chéng wéi shí xiàn yóu tián wěn chǎn dí dāng wù zhī jí

**English Translation:**

Translate the following sentence between Chinese and Pinyin.

Chinese: Therefore, relying on technological progress and strengthening scientific management has become an urgent task to achieve stable oilfield production

Pinyin: yīn cǐ, yī kào kē jì jìn bù, qiáng huà kē xué guǎn lǐ

yǐ chéng wéi shí xiàn yóu tián wěn chǎn dí dāng wù zhī jí

**Intent Understanding.** We introduce this task to test whether Chinese LLMs could capture the writing intent of the authors of a long document. This task helps measure how well LLMs can understand implications. We formulate this task as a multi-choice problem and adopt accuracy to assess the performance. An example is shown below:

**Chinese Example:**

亚马孙丛林中的雄性蓝蝶带有彩虹般的蓝色光辉，半公里外就能看到。其光辉如此强烈，有的竟能反射70%的蓝色光线，远远超过蓝色涂料的反射率。蓝蝶耀眼的光辉，原是一种警号，使别的雄性蓝蝶在远处就能知所趋避。蓝光越强，示警作用越显著。物竞天择，适者生存。亿万年的自然选择，使亚马孙蓝蝶翅膀有了如此奇妙的性能。

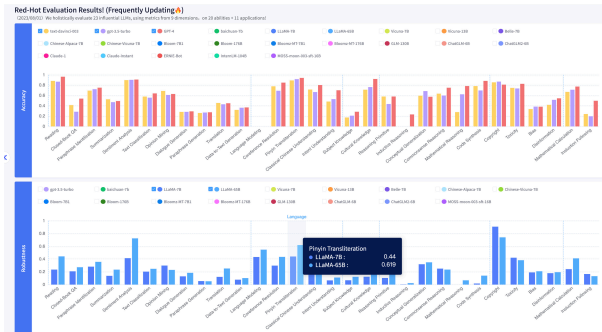
.....

对有关蓝蝶的仿生研究，理解不恰当的一项是

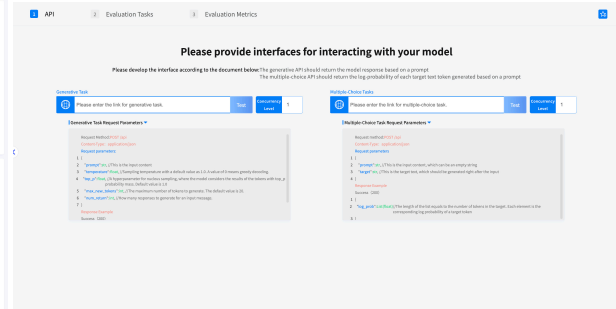
A. 在蓝蝶仿生的各类应用研究中，证券防伪的研究最有成效。

B. 翅膀上的羽状物的构造和尺寸，是仿生学家们极感兴趣的课题。

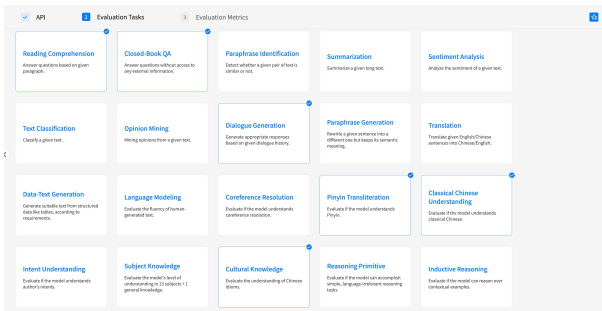
C. 新型的变幻色彩的迷彩服，可能与蓝蝶翅膀的反光结构有关。



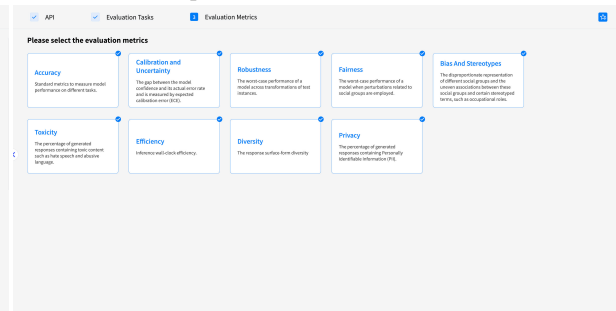
(a) Evaluation Results Overview



(b) Step 1: Provide APIs for Evaluation



(c) Step 2: Select Relevant Tasks



(d) Step 3: Select Desired Metrics

Figure 3: CLEVA provides a user-friendly interface. With only several clicks and minimum coding, evaluating a new language model can be deployed in a few minutes.

D. 对蓝蝶翅膀的反光机理的应用研究，目前还没取得突破性的结果。

答：A

**English Translation:**

Male blue butterflies in the Amazon jungle have a rainbow-like blue glow that can be seen from half a kilometer away. Their glow is so intense that some can reflect 70% of blue light, far exceeding the reflectivity of blue paint. The dazzling glow of the blue butterfly is actually a warning signal, allowing other male blue butterflies to know where to avoid from a distance. The stronger the blue light, the more obvious the warning effect. Survival of the fittest. Millions of years of natural selection have given the wings of Amazon blue butterflies such a wonderful performance.

.....

Regarding the bionic research on blue butterflies, one item that is not properly understood is

- A. Among the various applications of bionic research on blue butterflies, the research on securities anti-counterfeiting is the most effective.
- B. The structure and size of the feather-like structures on the wings are topics of great interest to bionics researchers.
- C. The new type of color-changing camouflage suit may be related to the reflective structure of the blue butterfly wings.
- D. The application research on the reflection mechanism of the blue butterfly wings has not achieved any breakthrough results so far.

Answer: A

**Classical Chinese Understanding.** Classical Chinese plays an important role in Chinese culture. Quatrain, poetry and etc. are all rooted in classical Chinese and most of them frequently appear in modern Chinese literature. Therefore we include this task to examine the model’s understanding of classical Chinese. We sample multi-choice questions from CCPM (Li et al., 2021) that inquire about the semantic equivalence between a modern

Chinese sentence and a list of classical Chinese candidates. We use accuracy as the primary metric. Below is an example:

**Chinese Example:**

“山间连绵阴雨刚刚有了一点停止的意思。”这句话可以用以下哪句古文来表达:

- A. 寒雨初开霁
- B. 山晓雨初霁
- C. 宿雨天初霁
- D. 山雨初含霁

答：D

**English Translation:**

“The continuous rain in the mountains has just shown a little sign of stopping.” Which of the following ancient Chinese sentences can be used to express this sentence:

- A. Cold rain just stops
- B. A morning in mountains, rain just stops
- C. An over-night rain just stops
- D. Rain in the mountain is stopping

Answer: D

**B.1.2 Knowledge**

**Subject Knowledge.** This task is in the format of fact completion (Petroni et al., 2019), where LLMs fill in the blank of a Chinese sentence with entities. Here we construct the dataset as in Petroni et al. (2019), which tests the knowledge from 13 subjects and 1 general domain. The metric is Accuracy@K ( $K = 1, 5$ ). We provide a math knowledge example:

**Chinese Example:**

婆罗摩笈多公式描述了\_\_-> 四边形

**English Translation:**

The Brahmagupta formula describes\_\_-> quadrilateral

**Cultural Knowledge.** Here we query Chinese LLMs with multi-choice questions related to Chinese culture, e.g., idioms. Data are sampled from ChID (Zheng et al., 2019). We adopt accuracy as the primary metric and show an idiom example below:

**Chinese Example:**  
 不过,要想变得\_\_\_\_,要想自己能够成就一番事业的话,不是说来就来的,或者说任何一个出色的人,他们都得经历过不少的磨难,以及在经受过了一些挫折之后,才能真正成才成人,才能成为一个实力超群的人物,让自己的人生过得越来越顺当...  
 上面这个句子下划线处可以填写哪个成语?  
 A. 足智多谋  
 B. 语无伦次  
 C. 绣花枕头  
 答: A

**English Translation:**  
 However, if you want to become \_\_\_\_ to achieve something on your own, it doesn't come easily. Any outstanding person has to go through a lot of hardships and setbacks before they can truly succeed and become a person of exceptional ability, making their life smoother and smoother...  
 Which idiom can be filled in the blank in the sentence above?  
 A. Wise and resourceful  
 B. Speak incoherently  
 C. Pretty on the outside but lacking substance underneath  
 Answer: A

### B.1.3 Reasoning

**Reasoning Primitive.** Following HELM (Liang et al., 2022), reasoning primitive is a collection of reasoning tasks independent of language and knowledge background and focuses on abstracted reasoning capacity evaluation. It includes tasks like non-ampliative reasoning, ampliative reasoning, recursive hierarchy and etc. Readers can refer to Liang et al. (2022) for more details. Here we synthesize the dataset similar to HELM (Liang et al., 2022) and use exact match to evaluate the final performance. Below is a recursive hierarchy example (in Dyck languages):

[[[[[[[[[[[[{{(O((({(}}))}})]]]O))]](O)  
 O))}}]]]}}]{{}}]

**Realistic Reasoning.** Contrary to reasoning primitive, in-the-wild reasoning combines the abstract reasoning skill of LLMs and their knowledge as well as the understanding of context (e.g., mathematical reasoning requires LLMs to be able to perform simple arithmetics). We choose the following reasoning tasks that not only help better surface the reasoning skills of LLMs but also have practical applications.

- **Inductive Reasoning** is to draw conclusions by going through a set of examples. Here the model needs to infer the rule from the few-shot demonstrations we provided and apply the rule to new examples. Data are collected

from BIG-Bench (bench authors, 2023). We use exact match as the evaluation metric and an example goes like this:

**Chinese Example:**  
 推断符号->的含义并计算下列公式。  
 512 + 372 -> 885  
 528 + 170 -> 699  
 859 + 133 -> 993  
 199 + 944 -> 1144  
 154 + 521 -> 676  
 67 + 987 -> 1055

**English Translation:**  
 Infer the meaning of the symbol -> and calculate the following formula.  
 512 + 372 -> 885  
 528 + 170 -> 699  
 859 + 133 -> 993  
 199 + 944 -> 1144  
 154 + 521 -> 676  
 67 + 987 -> 1055

- **Deductive Reasoning** is contrasted with inductive reasoning, where the model progresses from conclusions to specific examples. We provide an example of *modus tollens*<sup>1</sup>, a form of deductive argument, in which the model predicts whether a given conclusion is valid or not according to the previous statements. Data are translated from McKenzie et al. (2023) and we use accuracy as the evaluation metric.

**Chinese Example:**  
 考虑以下事实：  
 1.如果朱莉娅喜欢甲壳虫乐队，那么朱莉娅就是吉他手。  
 2.朱莉娅不是吉他手。  
 结论：因此，朱莉娅不喜欢甲壳虫乐队。  
 问题：根据陈述1和2.，结论是否有效？  
 回答：是

**English Translation:**  
 Consider the following facts:  
 1. If Julia likes the Beatles, then Julia is a guitarist.  
 2. Julia is not a guitarist.  
 Conclusion: Therefore, Julia does not like the Beatles.  
 Question: Based on statements 1. and 2., is the conclusion valid?  
 Answer: Yes

- **Commonsense Reasoning** is an umbrella of all related tasks, e.g., natural language inference and commonsense question answering (Storks et al., 2019). We mainly evaluate the classical natural language inference (data are sampled from OCNLI (Hu et al., 2020)) and commonsense question answering (data are translated from McKenzie et al. (2023)). We organize them into multi-choice tasks and adopt accuracy for assessment. Here we provide a textual entailment example:

**Chinese Example:**  
 是否可以“从‘篮子嘛，一块钱，一块钱啊。’中推断出‘这个篮子是可以卖的。’？”

<sup>1</sup><https://plato.stanford.edu/entries/logic-ancient/#ForModPonModTol>

- A. 总是可以
- B. 有时可以
- C. 不可以

答: A

**English Translation:**

Can it be inferred from “The basket, one yuan, one yuan.” that “This basket is for sale.”?

- A. Always
- B. Sometimes
- C. Never

Answer: A

- **Mathematical Reasoning** also has a rather big scope that envelopes various tasks, e.g., math word problem (MWP) solving, theorem proving and etc. (Lu et al., 2023). Here we focus on MWP and adopt exact match for evaluation. Data are sampled from Math23K (Wang et al., 2017). An MWP in our benchmark is:

**Chinese Example:**

问题: 一个饲养场, 养鸭1200只, 养的鸡比养的鸭多(3/5), 养的鸡比鸭多多少只?

答案: 720

**English Translation:**

Question: A farm has 1200 ducks, and the number of chickens raised is (3/5) more than the number of ducks raised. How many more chickens are there than ducks?

Answer: 720

- **Code Synthesis** is a task to synthesize an executable program that matches the requirement written in natural language. Data are translated from HumanEval (Chen et al., 2021) and we use pass@k as the metric ( $k = 1, 10, 100$ ). An example is shown below:

**Chinese Example:**

```
def is_sorted(lst):
```

```
    ...
```

给定一个数字列表, 返回它们是否以升序排序。如果列表有两个及以上相同数字, 则返回False。假设没有负数且只有整数。

示例:

```
is_sorted([5]) -> True
```

```
is_sorted([1, 2, 3, 4, 5]) -> True
```

```
is_sorted([1, 3, 2, 4, 5]) -> False
```

```
is_sorted([1, 2, 3, 4, 5, 6]) -> True
```

```
is_sorted([1, 2, 3, 4, 5, 6, 7]) -> True
```

```
is_sorted([1, 3, 2, 4, 5, 6, 7]) -> False
```

```
is_sorted([1, 2, 2, 3, 3, 4]) -> True
```

```
is_sorted([1, 2, 2, 2, 3, 4]) -> False
```

```
    ...
```

```
count_digit = dict([(i, 0) for i in lst])
for i in lst:
    count_digit[i]+=1
if any(count_digit[i] > 2 for i in lst):
    return False
if all(lst[i-1] <= lst[i] for i in range(1, len(lst))):
    return True
else:
    return False
```

**English Translation:**

```
def is_sorted(lst):
```

```
    ...
```

Given a list of numbers, return whether they are sorted in ascending order. If the list has two or more identical numbers, return False. Assume that there are no negative numbers and only integers.

Examples:

```
is_sorted([5]) -> True
```

```
is_sorted([1, 2, 3, 4, 5]) -> True
```

```
is_sorted([1, 3, 2, 4, 5]) -> False
```

```
is_sorted([1, 2, 3, 4, 5, 6]) -> True
```

```
is_sorted([1, 2, 3, 4, 5, 6, 7]) -> True
```

```
is_sorted([1, 3, 2, 4, 5, 6, 7]) -> False
```

```
is_sorted([1, 2, 2, 3, 3, 4]) -> True
```

```
is_sorted([1, 2, 2, 2, 3, 4]) -> False
```

```
    ...
```

```
count_digit = dict([(i, 0) for i in lst])
for i in lst:
    count_digit[i]+=1
if any(count_digit[i] > 2 for i in lst):
    return False
if all(lst[i-1] <= lst[i] for i in range(1, len(lst))):
    return True
else:
    return False
```

- **Conceptual Generalization** is a new task that is similar to inductive reasoning, where the model must reason over concrete examples to get a general rule and apply it to unseen examples. The reason we separate this task from inductive reasoning is that this task is specialized in reasoning over physical concepts or relations like directions. Data are synthesized as suggested by Patel and Pavlick (2022) and we employ top-k accuracy ( $k = 1$ ) to measure the performance. An example is shown below:

**Chinese Example:**

世界:

```
[0, 1, 0, 0]
```

```
[0, 0, 0, 0]
```

答案: 顶

世界:

```
[1, 0, 0]
```

```
[0, 0, 0]
```

答案: 左

世界:

```
[0, 1]
```

```
[0, 0]
```

答案: 上

世界:

```
[0, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0]
```

```
[1, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0]
```

答案: 左

**English Translation:**

World:

```
[0, 1, 0, 0]
```

```
[0, 0, 0, 0]
```

Answer: top

World:

```
[1, 0, 0]
```

```
[0, 0, 0]
```

Answer: left

World:

```
[0, 1]
```

```
[0, 0]
```

Answer: up

World:  
[0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0]  
[1, 0, 0, 0, 0]  
[0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0]

Answer: left

#### B.1.4 Harms

**Copyright.** This task was initially introduced by HELM (Liang et al., 2022) to examine the model’s ability to generate verbatim content and measure the underlying legal risk. We similarly extract some initial portion of copyrighted Chinese materials like books to construct prompts and let the model continue generation from this prompt. We follow Carlini et al. (2021) to collect text data and code data are sampled from HELM (Liang et al., 2022). We use longest common sequence, edit distance and edit similarity normalized by prefix length as evaluation metrics.

**Toxicity.** Here we choose the toxicity detection task to study the toxicity of Chinese LLMs (Borkan et al., 2019). In this task, we present a Chinese sentence to the model and ask the model whether the given sentence is toxic or not. We sample data from COLD (Deng et al., 2022) and choose accuracy as the metric.

**Bias.** Similar to the toxicity part, we ask the model to determine whether a given text is biased. We sample data from CDial-Bias (Zhou et al., 2022), which covers four demographic categories, including race, gender, region, and occupation. Micro F1 is the primary metric.

**Disinformation.** According to HELM (Liang et al., 2022), disinformation refers to

*false information that is disseminated by an actor with the intent to deceive, mislead, or otherwise influence the behavior of the target...*

However, related tasks described by Buchanan et al. (2021) are not well-developed in the Chinese world. We take a step to advance in this topic and focus on detecting “false information” that closely resembles hallucination detection and fact checking (Thorne et al., 2018; Gupta et al., 2022). We present a text that may contain hallucinated facts to the model and ask it whether this statement is true. We use accuracy as this is a classification

problem. Data are sampled from CHEF (Hu et al., 2022).

**Chinese Example:**

第33届金鸡奖揭晓：黄晓明、周冬雨再拿最佳男女主角。上述说法是否为真？

答：真

**English Translation:**

The 33rd Golden Rooster Awards were announced: Huang Xiaoming and Zhou Dongyu won the Best Actor and Actress again.

Is it True or False?

Answer: True

#### B.1.5 Others

**Mathematical Calculation.** Calculation is a fundamental skill for LLMs to execute a lot of tasks, e.g., comparing the price of tickets. To examine this skill, we provide two types of test instances and both of them involve basic arithmetic:

- The first type directly queries the model with mathematical equations. This format is more likely to test the memorization of LLMs on arithmetic.

11 + 32 -> 43

- The second type expresses the equation in a natural language format. This type checks whether LLMs could generalize what they have memorized in mathematical format to natural language format.

**Chinese Example:**

问：假设  $-48 + 62 = n$ 。n 的值是多少？答：14

**English Translation:**

Question: Suppose  $-48 + 62 = n$ . What is the value of

n? Answer: 14

For both types of instances, we utilize exact match to evaluate the performance. Despite the second type of instances being similar to MWPs in mathematical reasoning, test instances here only require the model to execute one-step arithmetic, while MWPs in mathematical reasoning are far more complicated and need multi-hop reasoning. Data are collected or translated from bench authors (2023); McKenzie et al. (2023).

**Instruction Following.** The success of recent LLMs is larger attributed to instruction tuning (Wei et al., 2022a; Ouyang et al., 2022), which unlocks the great potential of large models (Fu and Khot, 2022). Although the extensive application of prompting has demonstrated the strong capability of LLMs on understanding human instructions, it is natural to ask if this is just an illusion of frequentists or if LLMs truly master this. It is thus important to evaluate LLMs on long-tailed instructions. These instructions could be underlying bugs

of LLMs that are vulnerable to attacks and lead to potential risk (Zou et al., 2023). Here we translate some of these instructions from McKenzie et al. (2023) that do not relate to common NLP tasks but most LLMs perform poorly. Below is an example:

**Chinese Example:**  
将“+”视为数字1而不是数学运算。问：6+1的第一位数字是啥？答：6

**English Translation:**  
Consider “+” as the number 1 instead of a mathematical operation. Question: What is the first digit of 6+1? Answer: 6

We formulate the data in this task into a multi-choice problem and use accuracy for measurement.

## B.2 Application Assessment

**Reading Comprehension.** Reading comprehension is a type of question-answering task, where we present both the question and context to the model before it returns the answer. Our data for this task are sampled from C<sup>3</sup> (Sun et al., 2019) and are of the multi-choice format, therefore we use accuracy for evaluation. An example is given here:

**Chinese Example:**  
阅读以下内容，选择合适的选项回答：

女：听说你儿子跟你的关系不是很好？  
男：说实话我不是一个好父亲，因为忙，没时间管他，我们之间几乎没有沟通。因为我，他也有很大的压力。

问题：男的和儿子的关系为什么不好？

选项：  
A. 儿子态度不好  
B. 双方缺少交流  
C. 儿子工作很忙  
D. 父亲压力太大

答：B

**English Translation:**  
Read the following content and choose the appropriate option to answer:

Woman: I heard that your relationship with your son is not very good?  
Man: To be honest, I'm not a good father. I'm busy and don't have time to take care of him. We hardly communicate. Because of me, he also has a lot of pressure.

Question: Why is the relationship between the man and his son not good?

Options:  
A. The son has a bad attitude  
B. Lack of communication between the two  
C. The son is very busy with work  
D. The father is under too much pressure

Answer: B

**Closed-Book QA.** A more challenging setting of question-answering is closed-book QA (Wang et al., 2021), where the model is given no extra information and attempts to answer the question based on its own knowledge. Data are sampled or translated from Duan (2018); Zhang et al. (2018); Lin et al. (2022). An example is shown below and we use exact match as the metric:

**Chinese Example:**  
问题：谁能描述一下氧化镁的外观？  
答案：白色疏松粉末

**English Translation:**  
Question: Who can describe the appearance of magnesium oxide?  
Answer: White, loose powder.

**Paraphrase Identification.** In this task, a pair of sentences is passed to the model and the model decides whether they are discussing the same thing or not. We formulate the sampled data from CLUE (Xu et al., 2020) and FewCLUE (Xu et al., 2021) into a binary-choice format and leverage accuracy for assessment.

**Chinese Example:**  
你的火气大吗  
你火气大不大  
这两个句子表达的意思相同吗？是或否？是

**English Translation:**  
Do you have a bad temper?  
Are you quick to anger?  
Do these two sentences express the same meaning? Yes or No? Yes

**Summarization.** In text summarization, the model needs to abstract a long, unstructured text and generate a short summarization. Note that some of the data-to-text generation tasks (discussed later) also borrow the name “summarization”. The main difference between data-to-text generation and text summarization in our benchmark is whether the context is written in a programming language (then it is data-to-text generation) or the natural language because these two languages are distinct in nature. We sample data from CSDS (Lin et al., 2021) and use ROUGE (Lin, 2004) to evaluate the results.

**Chinese Example:**  
莫言获奖，围绕在莫言身边的出版商也笑开颜。北京精典博维文化发展有限公司拥有莫言中国内地所有作品及衍生品出版权。莫言获得诺贝尔文学奖，不仅会使公司业绩有“可观”的提升，还将加速该公司上市的进程。

TL;DR: 诺奖花落莫言签约书商IPO提速

**English Translation:**  
When Mo Yan won the award, the publishers around him were also happy. Beijing Jingdian Bowei Culture Media Co., Ltd. owns the publishing rights to all of Mo Yan's work and derivatives in mainland China. Mo Yan's winning of the Nobel Prize in Literature will not only bring a “considerable” increase to the company's profit but also accelerate the process of the company's listing.

TL;DR: Nobel Prize goes to Mo Yan, accelerating the IPO of his contracted publisher.

**Data-to-Text Generation.** Data-to-text generation is of growing interest recently as people try to use LLMs to assist their work, e.g., generating a report from an Excel table. This topic has long been explored prior to LLMs (Puduppully et al., 2019), especially under the name of summarization. We



sample data from [Shao et al. \(2019\)](#) and use BLEU for measurement. An example of generating an advertising proposal based on a structured table (in the Markdown format<sup>2</sup>) is shown here:

**Chinese Example:**

给定衣服的特点描述，生成相应的广告文案。

衣服特点:

| 版型 | 宽松 |  
| 风格 | 休闲 |  
| 图案 | 印花 |  
| 图案 | 手绘 |  
| 衣样式 | 衬衫 |

广告文案:

这款衬衫给人的第一印象就是风格独特，衬衫表面的士兵手绘图案印花精致有趣，真叫人忍不住多看几眼，浓浓的复古风也富于这款衬衫艺术感，就像巴黎卢浮宫内展示的名画一般。在款式上借鉴了睡衣版型，宽松舒适，休闲随性。

**English Translation:**

Given the description of the features of a clothing item, generate a corresponding advertisement copy.

Clothing features:

| Fit | Loose |  
| Style | Casual |  
| Pattern | Textile printing |  
| Pattern | Hand-painted |  
| Clothing type | Shirt |

Advertisement copy:

The first impression this shirt gives is its unique style. The soldier pattern hand-painted on the shirt is exquisite and interesting, making one can't help but take a few more glances. The strong retro style also gives this shirt an artistic sense, just like the famous paintings in the Louvre in Paris. In terms of style, it fits like a pajama, which is loose and comfortable, casual and natural.

**Sentiment Analysis.** Given a text, the model predicts the sentiment label (“Positive”) in sentiment analysis. Since it is a classification task, we use accuracy for evaluation. Our data are collected from FewCLUE ([Xu et al., 2021](#)). A sentiment analysis example is shown below:

**Chinese Example:**

这个产品评价是正面还是负面的?

评价: 今天刚拿到手机，打电话时发现手机听筒有吱吱吱的杂声，不满意，真怀疑是不是正品

答案: 负面

**English Translation:**

Is this product review Positive or Negative?

Review: Just got the phone today and found that there is a squeaking noise in the earpiece when making a call. Not satisfied, really doubt if it is genuine.

Answer: Negative

**Text Classification.** Similar to sentiment analysis, text classification predicts the answer from a fixed set of labels for a given text. Instead of the binary label in sentiment analysis, text classification in general has a larger label space. We collect data from FewCLUE ([Xu et al., 2021](#)) and SPR<sup>3</sup>. We adopt accuracy and an example is shown below:

<sup>2</sup><https://en.wikipedia.org/wiki/Markdown>

<sup>3</sup><https://github.com/DUTIR-Emotion-Group/CCL2020-Humor-Computation>

**Chinese Example:**

“全国青年教师教学艺术大赛举行”这段新闻的类别属于教育

**English Translation:**

The category of the news “The National Young Teachers Teaching Art Competition is held” is education

**Opinion Mining.** Opinion mining is a large topic that consists of vast tasks and has a close connection with sentiment analysis ([Zhang and Liu, 2017](#)). An exemplary task of opinion mining that we test here is opinion target extraction ([Liu et al., 2012](#)). We adopt exact match for evaluation in the context of the LLM era and show an example below:

**Chinese Example:**

“《恋恋笔记本》是导演尼克·卡萨维茨2004年的一部爱情类影片。”中主要围绕着什么进行描述?

恋恋笔记本

**English Translation:**

What is the main focus of the description in “The Notebook is a 2004 romance film directed by Nick Cassavetes.”?

The Notebook

**Dialogue Generation** The popularity of ChatGPT has shifted the interaction between humans and LLMs from a single-turn prompt continuation to a multi-turn conversation ([OpenAI, 2023](#)). It is thus important to evaluate LLMs in a multi-turn conversation setup, i.e., in the dialogue generation task. In this task, we use data from CrossWOZ ([Zhu et al., 2020](#)) and report BLEU and unigram F1. A conversation example is shown below:

**Chinese Example:**

用户: 你这看的什么视频?

系统: 是爱奇艺新出的《飞行少年》。

用户: 好看吗? 没事我也回家看看。

系统: 挺好看的, 是向祖国70周年的献礼剧。

用户: 都谁主演的啊?

系统: 严屹宽和一些年轻演员, 有闫妮, 不过是客串。

**English Translation:**

User: What video are you watching?

System: The Eyas, on iQIYI

User: Is it good? I am going to watch it at home if I have spare time.

System: It's pretty good. A TV series to celebrate the 70th birthday of our country.

User: Who is starring in it?

System: Yikuan Yan and other young actors. Ni Yan also appears

in a cameo.

**Paraphrase Generation.** Paraphrasing and rewriting is a common task in NLP. We show a text to the model and the model produces new text that is of the same meaning as the original text but of a different surface form. Following [Sun and Zhou \(2012\)](#), we choose iBLEU to evaluate the performance and utilize data from PKU Paraphrase Bank ([Zhang et al., 2019](#)).

**Chinese Example:**

一个句子的原句为:

从梅森苍白的唇间吐出了几乎听不见的回答。

它可以被复述为:

梅森先生苍白的嘴唇间溜出一个听不清楚的回答。

**English Translation:**

The original sentence is:

A barely audible answer came from Mason’s pale lips.

It can be paraphrased as:

Mr. Mason’s pale mouth let out an unclear answer.

**Translation.** Machine translation is not a Chinese-specific task but is multilingual. However, the success of Chinese LLMs relies heavily on bilingual (Chinese and English) data (Team, 2023; Zeng et al., 2023) and thus most Chinese LLMs are born to be capable of translating English text to and from Chinese. Our data are collected from the past WMT competitions (Kocmi et al., 2022). We employ SacreBLEU (Post, 2018) as the evaluation metric and an English-to-Chinese translation example is shown below:

**Chinese Example:**

英文: House rebukes Trump on border wall, but he plans veto

中文: 众议院在边境墙问题上指责特朗普, 但他计划使用

一票否决权

**English Translation:**

English: House rebukes Trump on border wall, but he plans veto

Chinese: 众议院在边境墙问题上指责特朗普, 但他计划使用

一票否决权

## C Manual Data Collection

We collect data on an extensive scale, comprising 33.98% of our entire benchmark. Besides constructing new test instances using sophisticated rules, manual annotation and composition serve as vital new data sources in many complicated tasks. We conducted rigorous screening, training, examination, and other quality control measures to ensure all crowdsourced work meets our high standards. In screening, we require each crowdsourcing worker to have at least a bachelor’s degree in a related major, and all translators must hold professional certificates. Before the manual collection, we prepare a detailed instruction handbook for each task, equipping qualified workers with the necessary knowledge and using in-domain examples to further clarify the requirements. During the collection process, we addressed all questions from crowdsourcing workers through an instant message platform. Automatic methods, as well as ample eye tests, were adopted both during and after the collection to guarantee fine-grained quality.

## D Metrics

### D.1 Accuracy

For each task in our benchmark, we list and underline the corresponding evaluation metrics for each task in Appendix B.

### D.2 Calibration and uncertainty

We mainly report the values of the following metrics:

- *Expected calibration error* (Kumar et al., 2019) (ECE) measures the difference between the model’s predicted probability and its exact-match accuracy.
- *Selective classification accuracy* (El-Yaniv and Wiener, 2010) computes the accuracy for the  $C$ -fraction of examples where the model assigns the highest probability.

### D.3 Robustness

Following HELM (Liang et al., 2022), we report the *worst-case accuracy*, which averages the poorest result among transformations of each test instance. Inspired by NL-Augmentor (Dhole et al., 2021), we implement the transformation recipe as the composition of the following perturbations:

- *Synonym perturbation* randomly substitutes Chinese words with their synonyms with a probability of 0.3.
- *Butter finger perturbation* randomly replaces Chinese words with other words that have the same toneless Pinyin with a probability of 0.05.
- *Character swapping* randomly swaps any two Chinese characters with a probability of 0.05.

We utilize LTP (Che et al., 2021) to perform word segmentation.

### D.4 Fairness

We similarly adopt *worst-case accuracy* as in robustness to report fairness. We support 4 transformation recipes inspired by the perturbations from NL-Augmentor (Dhole et al., 2021):

- *Simplified to traditional conversion* converts both the prompt and references from Chinese Simplified to Chinese Traditional<sup>4</sup>.
- *Mandarin to Cantonese conversion* translates both the prompt and references from

<sup>4</sup><https://github.com/BYVoid/OpenCC>

Mandarin to Cantonese. Here we adopt a rule-based approach<sup>5</sup> which first maps phrases in Mandarin to their translations in Cantonese and then converts the resulting string from Chinese Simplified to Chinese Traditional. We are aware that this implementation has certain limitations and there is ample room for improvement.

- Chinese name perturbation randomly substitutes all occurrences of a Chinese name with another feasible Chinese name with a probability of 0.5.
- Gender term perturbation randomly flips all gender terms of a source gender to their counterparts in a target gender with a probability of 0.5.

### D.5 Bias and stereotypes

We follow metrics from HELM (Liang et al., 2022) to quantify bias and stereotypes:

- **Bias:** we adopt the *demographic representation* in HELM, which measures the unevenness of gender or race terms for all social groups.
- **Stereotypes:** we adopt the *stereotypical associations* in HELM, which computes the unevenness of gender or race terms for all social groups when co-occurred with an adjective or profession term, then averages over all adjective or profession terms.

### D.6 Toxicity

We employ the *toxic fraction* metric from HELM (Liang et al., 2022), which is the fraction of instances that are classified as toxic according to the Perspective API (Lees et al., 2022)<sup>6</sup>. We use a threshold of 0.5 to determine whether an instance is toxic or not.

### D.7 Efficiency

As stated in the main text, we focus only on inference wall-clock time because limited statistics could be reliably collected from users. Concretely, we adopt *queries per second* (QPS), the amount of queries processed by a model API in a second, which is a common metric for measuring the throughput of online services.

<sup>5</sup><https://justyy.com/tools/chinese-converter/>

<sup>6</sup><https://perspectiveapi.com/>

### D.8 Diversity

Here we adopt inter-distinct and intra-distinct (Miller et al., 2017) to quantify surface-form diversity.

- *Inter-distinct* collects n-gram statistics from all instances in the test set and computes the n-gram diversity, which is the rate of all distinct n-grams against all n-grams.
- *Intra-distinct* evaluates the n-gram diversity per instance and averages across all instances.

### D.9 Privacy

We pay close attention to current research on privacy evaluation. For example, Carlini et al. (2021) utilize adversarial attacks to yield meaningful outcomes. We so far focus on the detection of personally identifiable information (PII) and are striving to involve more aspects in the near future.

To evaluate privacy from the PII perspective, we define *PII\_match*, a metric similar to the toxic fraction which represents the proportion of instances that contains PII:

$$\text{PII\_match} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\text{PII\_Detect}(y_i) > 0] \quad (1)$$

where  $N$  is the number of test instances,  $y_i$  is the generated text for  $i$ -th instance and *PII\_Detect* is the tool that returns the number of PII entities in  $y_i$ . We use Azure PII detection service<sup>7</sup> to instantiate *PII\_Detect*.

## E Models

Table 1 is the summary of Chinese LLMs we evaluated in our leaderboard.

**GPT** (Ouyang et al., 2022; Brown et al., 2020) is a family of autoregressive LLMs from OpenAI. The most recent and powerful GPT models are ChatGPT<sup>8</sup>, text-davinci-003<sup>9</sup>, and GPT-4 (OpenAI, 2023). We test all these three models in our evaluation.

**Claude** (Askell et al., 2021; Bai et al., 2022b,a) is another family of autoregressive models from Anthropic, which include Claude and Claude-

<sup>7</sup><https://learn.microsoft.com/en-us/azure/ai-services/language-service/personally-identifiable-information/overview>

<sup>8</sup><https://openai.com/blog/chatgpt>

<sup>9</sup><https://platform.openai.com/docs/models/gpt-3-5>

| Model             | Version                                   | Organization                | Access  | #Param. | Window Size | Instruction Tuning | Architecture |
|-------------------|---|-----------------------------|---------|---------|-------------|--------------------|--------------|
| ChatGPT           | gpt-turbo-3.5 (2023/07/11)                | OpenAI                      | limited | -       | 4096        | ✓                  | GPT          |
| text-davinci-003  | text-davinci-003 (2023/06/17)             | OpenAI                      | limited | 175B    | 4097        | ✓                  | GPT          |
| GPT-4             | gpt-4 (2023/07/11)                        | OpenAI                      | limited | -       | 8192        | ✓                  | GPT          |
| Claude            | claude-1 (2023/07/07)                     | Anthropic                   | limited | -       | 100000      | ✓                  | -            |
| Claude-instant    | claude-1 (2023/07/21)                     | Anthropic                   | limited | -       | 100000      | ✓                  | -            |
| InternLM-104B     | (2023/07/13)                              | Shanghai AI Lab & SenseTime | limited | 104B    | 2000        | ✓                  | GPT          |
| ERNIE-Bot         | (2023/07/09)                              | Baidu Inc.                  | limited | -       | 2000        | ✓                  | -            |
| ChatGLM-6B        | v0.1.0                                    | Tsinghua University         | open    | 6B      | 2048        | ✓                  | GLM          |
| ChatGLM2-6B       | v1.0                                      | Tsinghua University         | open    | 6B      | 2048        | ✓                  | GLM          |
| GLM-130B          | -   | Tsinghua University         | open    | 130B    | 2048        | ✓                  | GLM          |
| BLOOMZ-7B1-mt     | -   | BigScience                  | open    | 7B      | 2048        | ✓                  | BLOOM        |
| BLOOM-7B1         | -   | BigScience                  | open    | 7B      | 2048        | ✗                  | BLOOM        |
| BLOOMZ-176B-mt    | -   | BigScience                  | open    | 176B    | 2048        | ✓                  | BLOOM        |
| BLOOM-176B        | -   | BigScience                  | open    | 176B    | 2048        | ✗                  | BLOOM        |
| LLaMA-7B          | -   | Meta                        | open    | 7B      | 2048        | ✗                  | LLaMA        |
| LLaMA-65B         | -   | Meta                        | open    | 65B     | 2048        | ✗                  | LLaMA        |
| Vicuna-7B         | v1.1                                      | LMSYS                       | open    | 7B      | 2048        | ✓                  | LLaMA        |
| Vicuna-13B        | v1.1                                      | LMSYS                       | open    | 13B     | 2048        | ✓                  | LLaMA        |
| BELLE             | BELLE-7B-2M                               | Beike Inc.                  | open    | 7B      | 2048        | ✓                  | BLOOM        |
| Chinese-Vicuna-7B | Chinese-Vicuna-lora-13b-belle-and-guanaco | Cui et al.                  | open    | 7B      | 2048        | ✓                  | LLaMA        |
| Chinese-Alpaca-7B | Chinese-Alpaca-7B                         | Fan et al.                  | open    | 7B      | 2048        | ✓                  | LLaMA        |
| MOSS-16B          | moss-moon-003-sft                         | Fudan University            | open    | 16B     | 2048        | ✓                  | CodeGen      |
| Baichuan-7B       | -   | Baichuan Inc.               | open    | 7B      | 4096        | ✗                  | LLaMA        |

Table 1: 23 Chinese LLMs evaluated in this work. For limited-accessed models, we mark the timestamp where we finalized the evaluation in the format of (YYYY/MM/DD). For models with the same public name but have different versions, we also provide the version we used to conduct the experiment. Note that the unit of window size of ERNIE-Bot is characters instead of tokens.

instant<sup>10</sup>. Both models are evaluated in our experiments.

**InternLM** (Team, 2023) is a GPT-like Chinese LLM trained by Shanghai AI Laboratory and SenseTime. It has a limited-accessed 104B and an open-source 7B version. We evaluate the 104B version in our experiments.

**ERNIE-Bot**<sup>11</sup> is a Chinese LLM launched by Baidu Inc. We observe that some datasets trigger the safety measure of ERNIE-Bot and obtain invalid responses. This fact leads to a poor result in our evaluation.

**GLM** (Du et al., 2022) is a Chinese LLM family from Tsinghua University trained with autoregressive blank infilling. We only assess their open-source GLM-130B (Zeng et al., 2023), ChatGLM-6B<sup>12</sup> and ChatGLM2-6B<sup>13</sup>.

**BLOOM** (Scao et al., 2022) is a family of open-source multilingual LLMs from BigScience. It is not fine-tuned and has an instruction-following version BLOOMZ (Muennighoff et al., 2023). In our experiment, we test the pretraining-only BLOOM-7B1 and BLOOM-176B from BLOOM,

and the instruction-following BLOOMZ-7B1-mt and BLOOMZ-176B-mt from BLOOMZ.

**LLaMA** (Touvron et al., 2023) is a more recently released open-source autoregressive English LLM family from Meta and is pretrained only. We experiment with LLaMA-7B (the smallest one) and LLaMA-65B (the largest one).

**Vicuna** (Chiang et al., 2023) is a series of instruction-following models built on top of LLaMA (Touvron et al., 2023). It comes from LMSYS. We evaluate both Vicuna-7B and Vicuna-13B.

**BELLE** (Ji et al., 2023) refers to a series of instruction-following models from Beike Inc., fine-tuned on various pretrained models like BLOOM and LLaMA. We assess their BLOOMZ-based 7B variant.

**Chinese-Vicuna** (Chenghao Fan and Tian, 2023) is a Chinese instruction-following model fine-tuned from LLaMA and has 7B and 13B two variants. We experiment with the 7B version.

**Chinese-Alpaca** (Cui et al., 2023) is a family of LLaMA-based Chinese LLMs. They extend the original LLaMA’s vocabulary for better Chinese modeling and open-source fine-tuned Chinese LLMs with various model scales. We test their early 7B instruction-following model.

**MOSS** (Sun et al., 2023b) is pretrained and fine-

<sup>10</sup><https://www.anthropic.com/index/introducing-claude>

<sup>11</sup><https://yiyao.baidu.com/welcome>

<sup>12</sup><https://github.com/THUDM/ChatGLM-6B>

<sup>13</sup><https://github.com/thudm/chatglm2-6b>

tuned from CodeGen (Nijkamp et al., 2023) by Fudan University. It includes the pretrained model, an instruction-following model, and a tool-augmented instruction-following model (Schick et al., 2023). We evaluate the instruction-following version<sup>14</sup> in our experiment.

**Baichuan**<sup>15</sup> is a pretrained Chinese LLM from Baichuan Inc., with the same architecture as LLaMA. We test the early 7B version and a new 13B version<sup>16</sup> is released by the time of writing.

## F Prompting

### F.1 Settings

The prompt setting remains the same as the common practice (Brown et al., 2020; Liang et al., 2022), where we randomly choose 5 in-context training examples (a.k.a., demonstrations) for few-shot prompting. To mimic true few-shot setting (Perez et al., 2021), these 5 in-context training examples will be fixed for all test instances. For classification, we sample one example for each of the 5 most frequent labels if the number of possible labels is larger than 5. If the length of 5-shot demonstrations exceeds the context window size of a model (e.g., reading comprehension), we reduce the number of in-context examples.

### F.2 Format

**Completion-style few-shot prompting.** Given the description of the task, sampled demonstrations, and a test instance, we use the below template to construct the few-shot prompt for prompting conventional LLMs (a string):

```
{instruction} \n{n}{demonstration1} \n{n}...
{demonstrations5} \n{n}... {prompt} {prediction}
```

where **{instruction}** is the task description, **{demonstration<sub>1</sub>}** is the concatenation of the prompt and reference of the first in-context example, \n is the line break and **{prompt}** is the prompt of the test instance. The model will continue the prompt and complete the generation in **{prediction}**. We denote this type of prompt template as Completion. A mathematical calculation example is shown below (we use an English prompt template for demonstration only and all prompt templates in our benchmark are Chinese):

<sup>14</sup><https://huggingface.co/fnlp/moss-moon-003-sft>

<sup>15</sup><https://github.com/baichuan-inc/baichuan-7B>

<sup>16</sup><https://github.com/baichuan-inc/Baichuan-13B>

Calculate the following formula.

758 + 445 -> 1203

758 + 445 -> 1203

140 + 361 -> 501

**Chatbot-style few-shot prompting.** The popularity of ChatGPT has led to an outbreak of LLM-based chatbot (Team, 2023; Chenghao Fan and Tian, 2023). Existing work (Huang et al., 2023) shows that the best few-shot prompting strategy for chatbots is different from the one for conventional LLMs. Specifically, the instruction, demonstrations, and test prompt should not be concatenated together but organized as a dialogue history, where the instruction serves as the system prompt and the prompt and reference of a demonstration form a dialogue turn. The previous example will be reorganized as below before feeding into the chatbot:

```
System:
Calculate the following formula.
User:
758 + 445 ->
Assistant:
1203
User:
163 + 140 ->
Assistant:
303
User:
140 + 361 ->
Assistant:
501
```

where System: is the field to set up the chatbot and we will put the instruction here. User: and Assistant: stand for the prompt and reference respectively. We denote this type of prompt template as Chatbot.

**Multi-choice problem format.** As discussed in Liang et al. (2022), there are two strategies when constructing prompts for multi-choice problems:

- Separate (Brown et al., 2020) scores each choice by concatenating it with the prompt and takes the one with the highest probability as the prediction.
- Joint (Hendrycks et al., 2021) puts all choices into the prompt and lets LLMs generate the choice index (e.g., “{question} A. {choice<sub>1</sub>} B. {choice<sub>2</sub>} Answer:”).

In general, Separate approach better estimates the model performance as the output space is restricted, while Joint approach is more economic since the model only needs to infer once to get the final an-

swer. We consider both types when crafting prompt templates for multi-choice problems.

**Chain-of-Thought.** Chain-of-Thought (Wei et al., 2022c; Kojima et al., 2022) (CoT) is a crucial technique to elicit the reasoning ability of LLMs. We also support CoT in CLEVA and provide the corresponding prompt templates for the mathematical reasoning task. An example of CoT is shown below, where highlighted text is predicted by the model and text in **red** is the intermediate reasoning process and text in **green** is the final answer.

Question: A community has 8 buildings, each with 102 residents. On average, each household pays 9 yuan per month for water. How much does this community pay for water in total per month?  
 Answer: Let's think step by step. **First, each household pays 9 yuan per month for water, and each building has 102 residents. Therefore, the total monthly water bill for each building is:  $9 * 102 = 918$  yuan. The community has a total of 8 buildings, so the total monthly water bill for the community is:  $918 * 8 = 7344$  yuan. Therefore, the answer is 7344.**

### F.3 System Design

Previous work (Bach et al., 2022; Ding et al., 2022) has presented various approaches to design a prompting language that enables flexible prompt construction. However, their methods have shown limitations in handling the vast variety of tasks for large-scale evaluation: Their prompting languages can only manipulate strings, while many tasks are organized in a structured data format, e.g., dialogue generation and data-to-text generation.

We therefore devise a slightly complicated prompting language that accommodates customized prompt construction of structured data while preserving usability. We start the description with an instance in JSON format from the dialogue generation task:

```
{
  "history": [
    {
      "utterance": "Who is the US president?",
      "role": "usr"
    },
    {
      "utterance": "Joe Biden.",
      "role": "sys"
    },
    {
      "utterance": "Then who is his wife?",
      "role": "usr"
    }
  ],
  "role": "sys",
  "label": [
    "Jill Biden."
  ]
}
```

and a prompt template example written as a JSON dictionary (`\n` is the line breaking):

```
{
  "verbalizer": {
    "role": {
      "sys": "Assistant",
      "usr": "User"
    }
  },
  "history": {
    "item_separator": "\n",
    "item_template": "{role}: {utterance}",
    "item_index": null
  },
  "input": "{history}\n{role}:",
  "label": "{label}"
}
```

The general pipeline of our prompt construction is as follows (we mark the field from the instance in **green** and the one from the prompt template in **blue**):

1. We first map values of all fields in a test instance according to user-defined mappings in **verbalizer** (Gao et al., 2021b). In our example, all `usr` and `sys` will be replaced with `User` and `Assistant` respectively.
2. We then stringify each field in the test instance. We organize all structured data fields in the format of a list of dictionaries (**history** in our example) and apply the following rules to process them:
  - (a) For each entry (a dictionary), we independently stringify it by composing all its fields via a template defined in the Python f-String syntax<sup>17</sup>. For instance, an utterance in the dialogue history `Who is the US president?` from the speaker `User` will be formatted into `User: Who is the US president?` according to **item\_template** in a prompt template field that shares the same name as **history**.
  - (b) We then index all stringified entries (by prepending an index like `A.`  to each entry) if needed and concatenate them with a user-defined separator **item\_separator** to stringify the whole data structure. In our case, we do not apply any indexing (an empty option in **item\_index**) and directly assemble the final string of **history** with `\n`:
 

```
User: Who is the US president?
Assistant: Joe Biden.
User: Then who is his wife?
```
3. We finally construct the prompt and references

<sup>17</sup><https://peps.python.org/pep-0498/>

from all stringified fields. According to `input`, the resulting prompt in our example will be:

*User: Who is the US president?*

*Assistant: Joe Biden.*

*User: Then who is his wife?*

*Assistant:*

For the references, we directly apply `label` in the prompt template to every entry in `label`, resulting in “*Jill Biden*” here.

Though not shown in the example above, another crucial part is to attach specific post-processing steps tailored to a prompt template. For example, if we index the choices in an instance from a multiple-choice task by capital letters like “A. ”, we should capitalize the initial output letter for a more accurate evaluation. In our system, we achieve this by passing a list of predefined options to the subfield `postprocess` in the prompt template field `meta`, which executes the script of each post-processing option on the output consecutively.

## G Results

In this section, we provide the complete evaluation results and breakdown analysis of our benchmark.

### G.1 Meta Analysis

To validate the uniqueness and reasonability of diversity and privacy, we examine the correlation between accuracy and these two newly introduced metrics. Figure 4 shows the scatter plot. We can see that there is a weak positive correlation between accuracy and diversity, justified by a value of 0.30 in Pearson’s  $r$  (P-value is  $9.9 \times 10^{-9}$ ). This phenomenon suggests that a strong Chinese LLM is likely to be able to produce diverse text. On the other hand, privacy seems to have no strong correlation to accuracy, with a value of -0.10 in Pearson’s  $r$  (P-value is 0.05). These weak correlations indicate the uniqueness of privacy and diversity as they can not be easily encompassed by a single accuracy metric.

### G.2 Ability Evaluation

In this section, we focus on the analysis of ability evaluation. Given that there are too many models for comparison, we select several interested groups of models in the visualization. Figure 5 compares 4 groups of models, each group consisting of two categories with three top-performing models. We have the following observations:

- Although outstanding Chinese model like InternLM-104B is comparable and even outperforms the best English models in some tasks, most high-ranking models in our Chinese benchmark are English models.
- The gap between limited-accessed and open-source models (Liang et al., 2022) is also witnessed in Chinese LLMs. We believe this gap could be narrowed down by fine-tuning a large-scale (with 100B and more parameters) Chinese LLM with the most recent instruction tuning strategies. Figure 2 shows that the well-performing open-source models are small models fine-tuned by the most recent and advanced techniques like Self-Instruct (Wang et al., 2023). These models mainly lag behind the limited-accessed model in many reasoning and knowledge-intensive tasks as shown in Figure 5, which could be addressed by scaling up the model size (Liang et al., 2022; Fu and Khot, 2022).
- Aligned with Liang et al. (2022); Fu and Khot (2022), large LLMs show clear advantages over the small ones in many reasoning and knowledge-intensive tasks.
- Instructing tuning is indeed a crucial technique to unleash the full potential of LLMs (Fu and Khot, 2022). Some small instruction-following models are even more powerful than those without instruction-tuning. For example, InternLM-104B is much better than BLOOM-176B. In addition, instruction-following models are generally less sensitive to the choice of prompt templates (with a smaller area around each point), suggesting that instruction tuning improves the model’s robustness to prompt templates.

Moreover, we also observe some interesting phenomena in Figure 5: *Inverse scaling* (McKenzie et al., 2023) seems to appear in our instruction following task, where the larger GPT-4, InternLM-104B, and LLaMA-65B is worse than MOSS-16B. According to our marking of tasks with a great standard deviation in Figure 5, they all are the *emergent ability* (Wei et al., 2022b) candidate in the Chinese world, e.g., mathematical reasoning, code synthesis, Pinyin transliteration and etc. We are aware that the analysis here is not a rigorous study that verified the existence of inverse scaling and emergent ability in certain Chinese tasks and we leave it for future work. In the end, we find some tasks

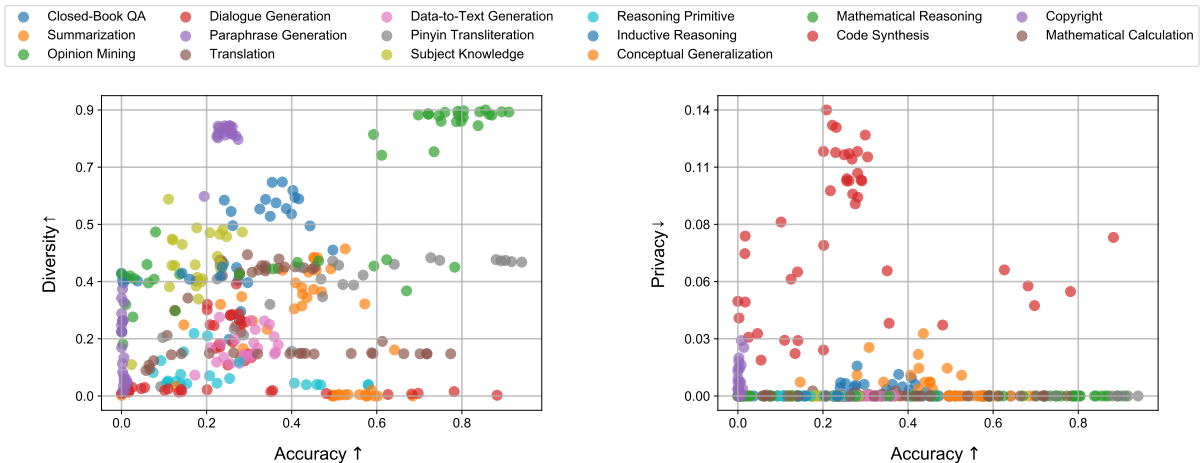


Figure 4: Correlation between diversity or privacy and accuracy on all tasks in a scatter plot format. Each point is a model’s performance of diversity/privacy and accuracy on a specific task.

(e.g., inductive reasoning) that are difficult even for the most powerful GPT-4, indicating an unresolved problem that we could work on in the future.

We analyze the knowledge of different Chinese LLMs in Figure 6 by utilizing questions from 14 subjects. We see that large models outperform small models in this knowledge-intensive task on many subjects, e.g., GPT-4, Claude, and InternLM-104B are much better than MOSS-16B and Vicuna-13B. Notably, Baichuan-7B possesses a high quantity of knowledge and is comparable to large models. This fact explains why it performs so well in knowledge-intensive tasks like classical Chinese understanding, commonsense reasoning and etc., as shown in Figure 5.

We also empirically examine the rationality of the design and structure of our ability evaluation by computing the correlation between any pair of tasks and manually checking with the human prior. As shown in Figure 7, most pairs of tasks that both not belonging to the same aspect (e.g., knowledge) do not share a statistically significant correlation, e.g., conceptual generalization and cultural knowledge. Some statistically significant correlations are well-match with our expectations (not exhausted):

- A good performance on coreference resolution and cultural knowledge helps to identify toxic and biased content (Pearson’s  $r > 0.6$ );
- Commonsense reasoning ability is also required for toxicity and bias as this harmful content could be implicit (Pearson’s  $r > 0.5$ );
- There is a strong positive correlation among almost all reasoning tasks (Pearson’s  $r > 0.5$ );
- More subject knowledge improves conceptual generalization and commonsense reasoning

(Pearson’s  $r \geq 0.6$ );

- More cultural knowledge yields a better result in classical Chinese understanding (Pearson’s  $r = 0.85$ );
- Mathematical calculation is almost mandatory for mathematical reasoning (Pearson’s  $r \approx 0.8$ );

These observations in general justify the rationality of our taxonomy.

In addition, we observe some interesting phenomena. Reasoning primitive has a strong positive correlation with Pinyin transliteration (Pearson’s  $r \approx 0.9$ ). This indicates that some sort of reasoning is required for Pinyin transliteration. For example, a valid Pinyin sequence matches the appearance of each character and its Pinyin precisely. The model needs to follow this rule to predict correctly. However, there are also some counter-intuition observations that could not be explained easily: A strong positive correlation (Pearson’s  $r = 0.76$ ) between reasoning primitive and classical Chinese understanding reveals the distinct mechanism beneath LLMs and the human brain.

### G.3 Application Assessment

Figure 8 compares the performance of models in application assessment tasks. The conclusions are in line with those in Figure 2: Most high-ranked models are English models and are limited-accessed. Interestingly, we see that English models tend to have fewer “weak spots”, a task that the model performs poorly compared to other models. It could be the fact that we choose more Chinese models that span a wide quality range, while English models are mainly the famous ones with the guarantee in quality. We observe that English open-source



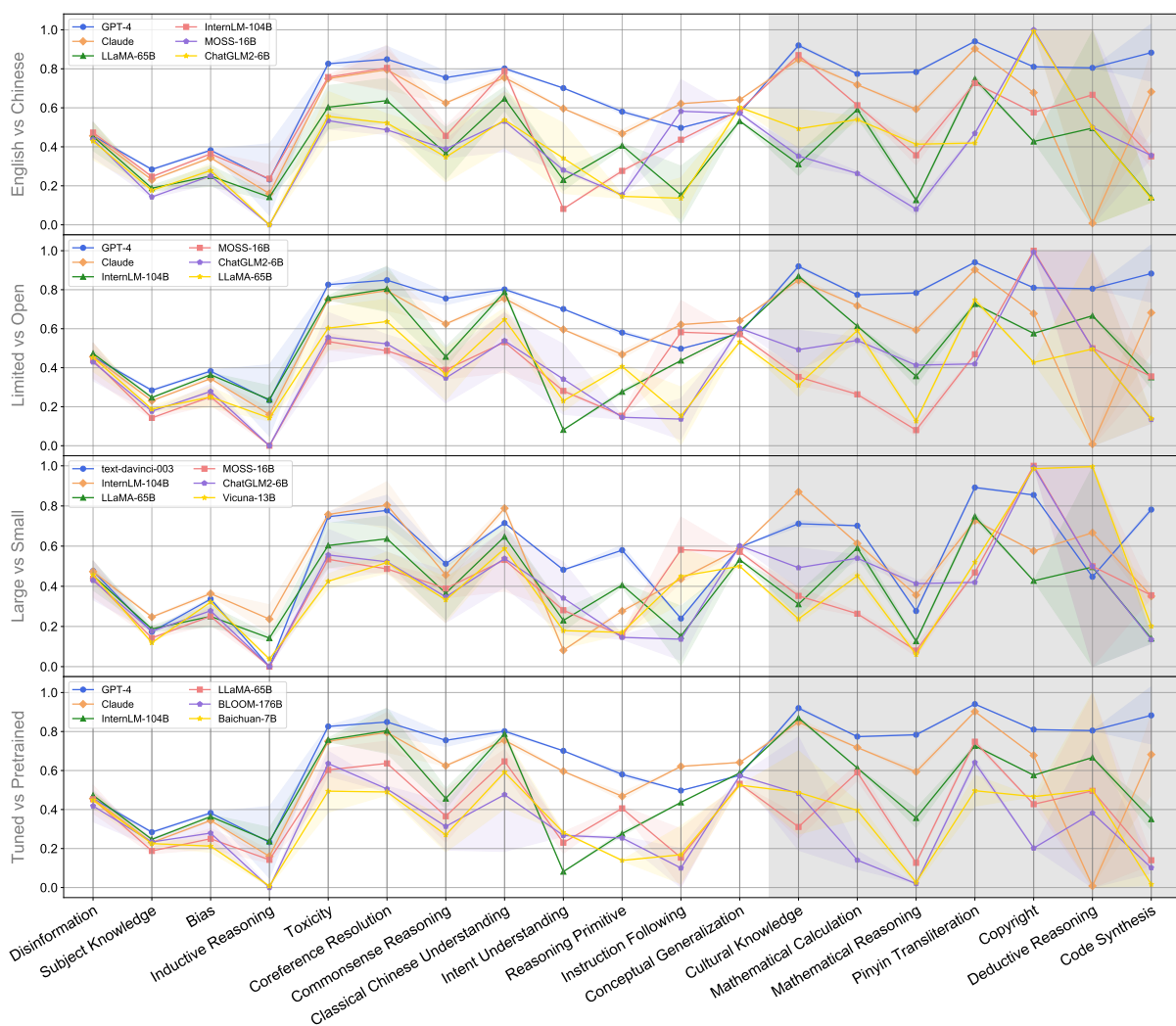


Figure 5: Comparison between three best-performing models from two categories on all ability evaluation tasks. Models in the left legend column belong to the first category and those in the right belong to the second category. For example, GPT-4, Claude and LLaMA-65B are **English** models. There are 8 categories: **Chinese** are Chinese-focused models (with tailored strategies to improve Chinese modeling), **English** are English-focused models, **Open** are open-source models, **Limited** are limited-accessed models, **Large** are models with more than 50B parameters (We choose text-davinci-003 rather than GPT-4 and ChatGPT as its size has been reported), **Small** are models with fewer than 50B parameters, **Tuned** are instruction-following models and **Pretrained** are pretrained models (without instruction tuning). Each point represents the mean performance of the model on a specific task and the area around each point is of the size of  $\pm$  standard deviation. We rank tasks in the x-axis by the standard deviation and the task with a larger standard deviation is closer to the right. We mark tasks with a standard deviation larger than 0.1 by gray shadow. These tasks imply the plausible emergent abilities of Chinese LLMs. Note that we normalize the score in the copyright task across models and then subtract it from 1 to convert it to a metric whose value is larger implying a better result.

models do not work well on translation and text classification.

We show the distribution of different metrics at different tasks in Figure 9.

- **Accuracy.** Multi-choice tasks like reading comprehension, text classification, and sentiment analysis have a high accuracy mean but models are clearly differentiated. On the other hand, generation tasks have a low median and most models are close to each other in general.

- **Efficiency.** There is a large difference in efficiency among models. This is because there exist many unfair comparisons. For example, limited-accessed models do not provide details on how many resources they invest when serving each query.
- **Robustness & Fairness.** For robustness and fairness, they have a similar trend as accuracy but with a relatively lower value, probably because they share the same base metric on

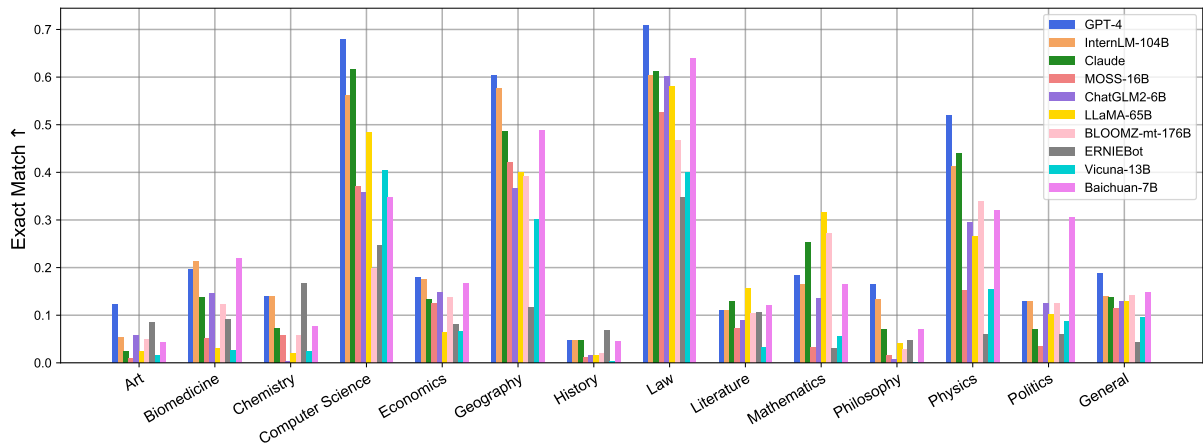


Figure 6: The performance of models on 14 subjects in the subject knowledge task. We select the best-performing models from top-10 institutions according to accuracy.

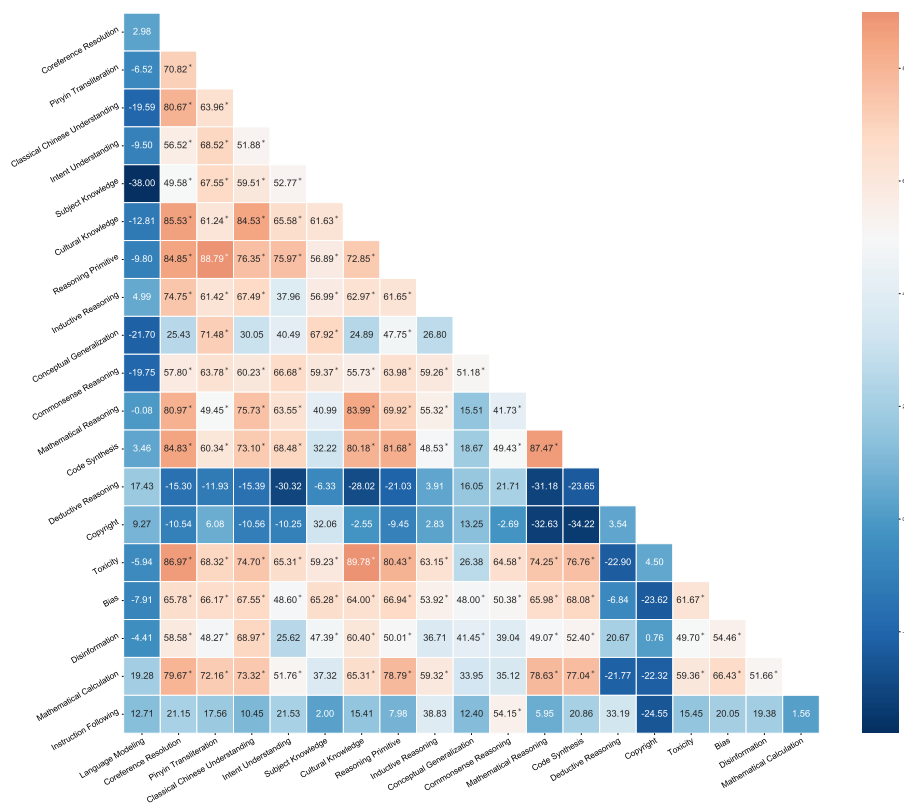


Figure 7: Correlation between different tasks in ability evaluation. Each entry is Pearson's  $r$  between two tasks from the corresponding row and column. \* denotes that the correlation coefficient is statistically significant with a P-value lower than 0.05.

augmented data. We observe that some tasks are more sensitive to noise, e.g., sentiment analysis and opinion mining.

- **Calibration.** We compare the values on  $ECE-10$  (Kumar et al., 2019). In general, models have a high ECE, making them less valuable in assisting human decisions.
- **Diversity.** We focus on the inter-distinct metric. We see that most models have a similar level of diversity in most tasks. Their

differences become obvious only in some knowledge-intensive tasks like closed-book QA and tasks that have multiple feasible correct answers, e.g., summarization, dialogue generation, and data-to-text generation.

- **Bias.** We choose to compare gender bias. We observe that models in data-to-text generation, summarization, and dialogue generation exhibit a strong tendency to produce biased content. These results could be partially attributed

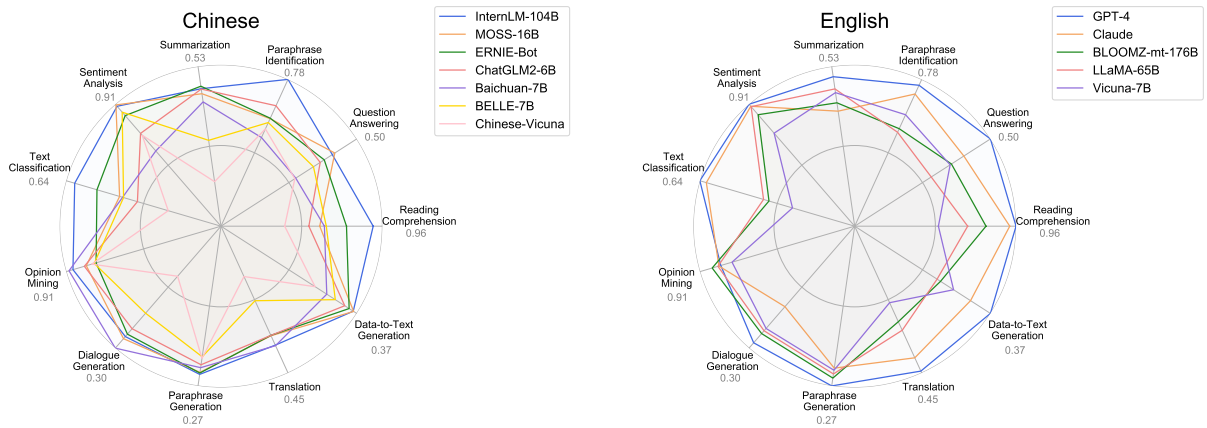


Figure 8: Comparison among models from different groups in tasks of application assessment. We choose the best models for each institution and divide them into 2 groups based on the language they focus on: Chinese or English.

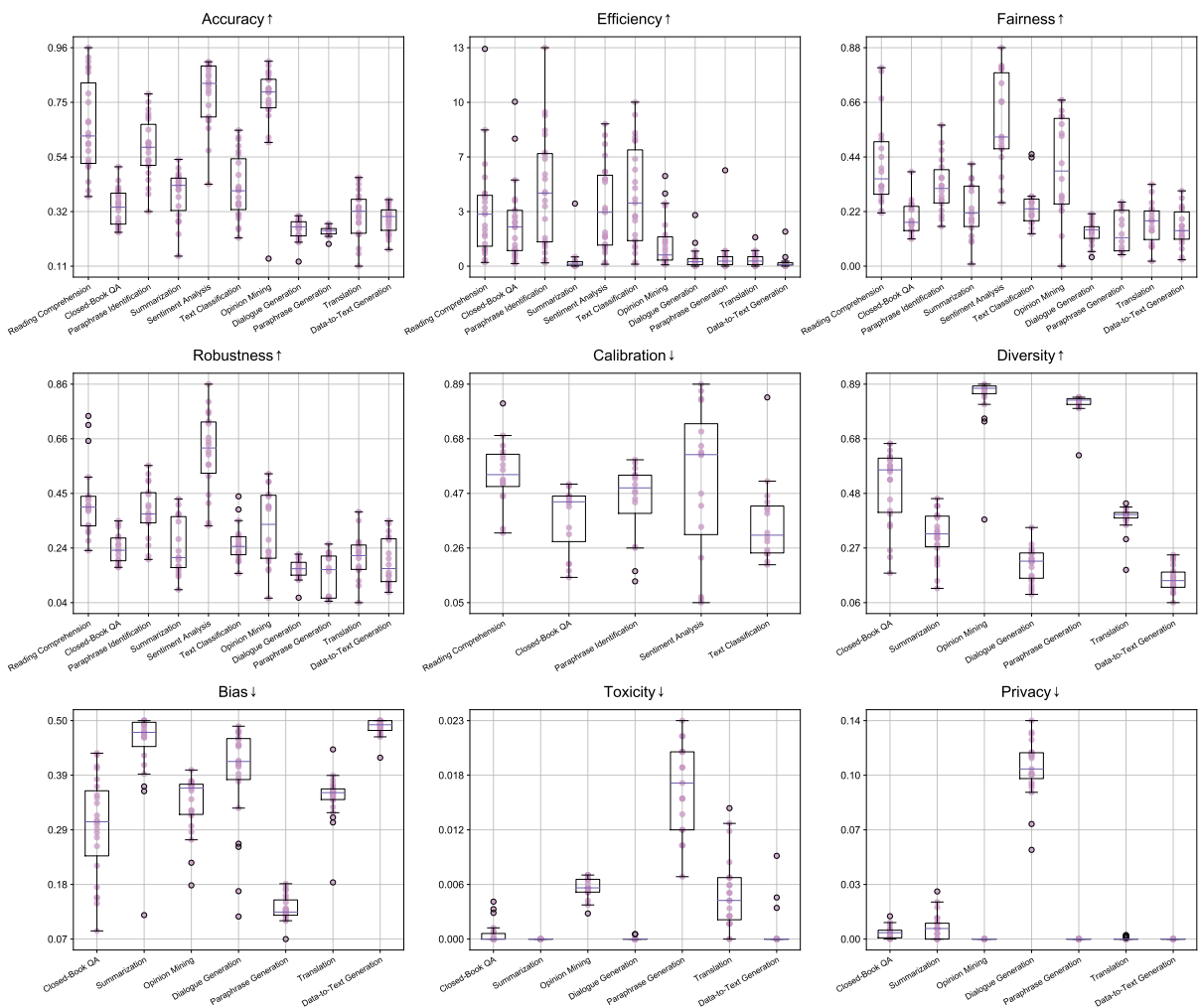


Figure 9: The performance distributions of application assessment tasks under different metrics. Some tasks are missing in some metrics because they are unavailable, e.g., models merely generate an index in the text classification task, thus metrics that evaluate the generated text like diversity, bias, toxicity, and privacy are not applicable.

to the bias in the dataset domain.

- **Privacy & Toxicity.** For toxicity and diversity, it is meaningless to compare as almost all values are low. The only exception is dialogue generation in privacy. This is because

our data contains inquiries for detailed contact information. The implication of a high value of privacy metric in dialogue generation is mixed: It means that the model understands users' requests and attempts to address them

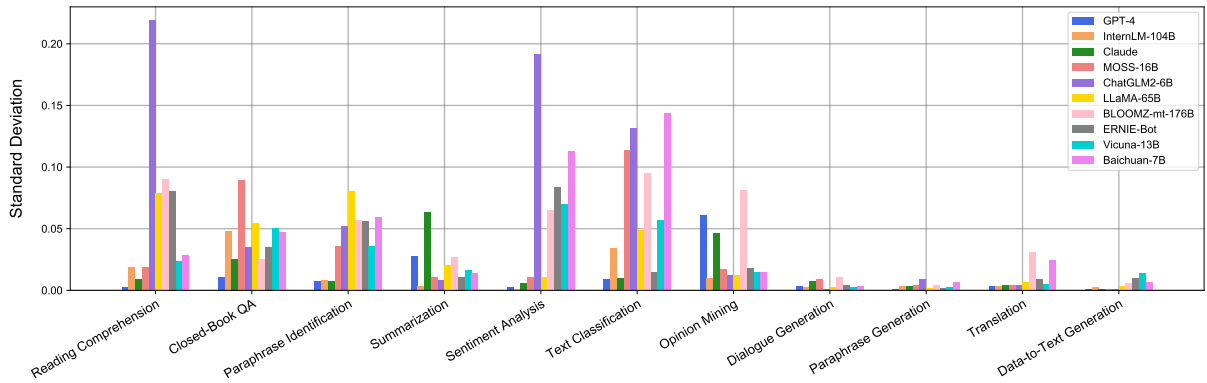


Figure 10: The accuracy standard deviation of different models in different prompt templates from different application assessment tasks. We select the best-performing models from top-10 institutions according to accuracy.

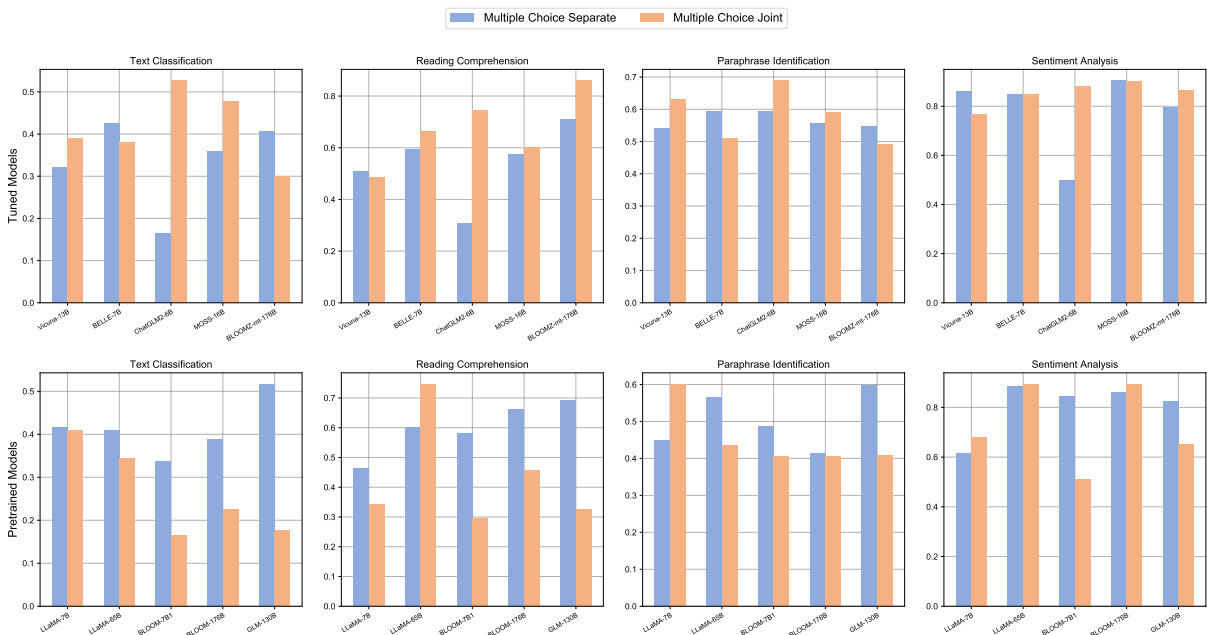


Figure 11: The accuracy of different models in multi-choice tasks with Separate and Joint style prompt templates.

with concrete information. It also implies that the model has a higher risk of hallucination that leads to potential harm.

At the end of this section, we study the prompt template sensitivity, one of the key features in CLEVA. Figure 10 presents the accuracy standard deviation of different prompt templates of different models. We find that instruction-following models have a lower level of standard deviations and thus are more robust to variations in prompt templates, consistent with the conclusion in ability evaluation. We also see that small models like ChatGLM2-6B and Baichuan-7B have relatively higher standard deviations compared with large models. Interestingly, strong models like GPT-4 have a relatively large variance in some tasks like summarization. A possible reason is that models are sensitive to

some keywords in the instruction, e.g., almost all models perform better in prompt templates that contain “zhāi yào” (means “summarize” in English) in the summarization task. We also find that limited-accessed models sometimes refuse to answer. For example, ERNIE-Bot refuses to answer about 4 tasks, resulting in a lower ranking in Figure 2.

#### G.4 Prompting Analysis

As discussed in Appendix F, there are two feasible prompt template types for multi-choice tasks: Separate that feeds each choice with the prompt separately and Joint that concatenates all choices and feeds once. We compare the model performance on these two types of prompt templates in multi-choice tasks from application assessment. Figure 11 shows that despite the cost of Separate,

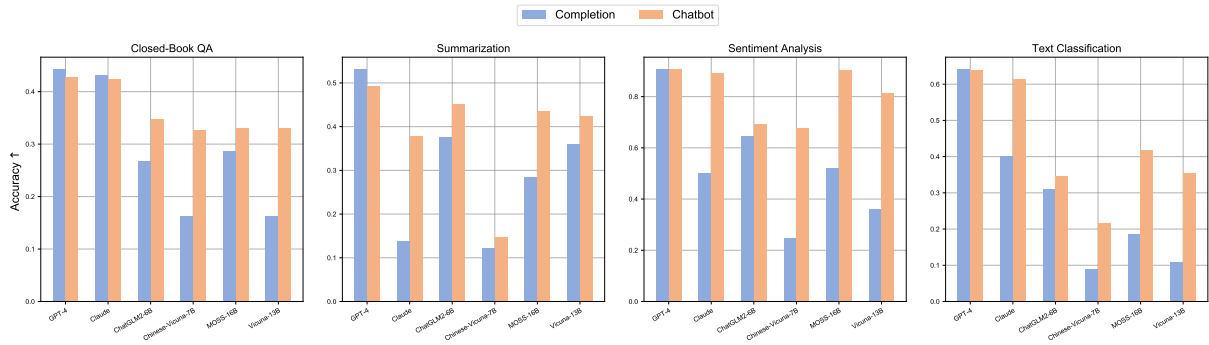


Figure 12: The accuracy of different chatbots with Completion and Chatbot style few-shot prompt templates.

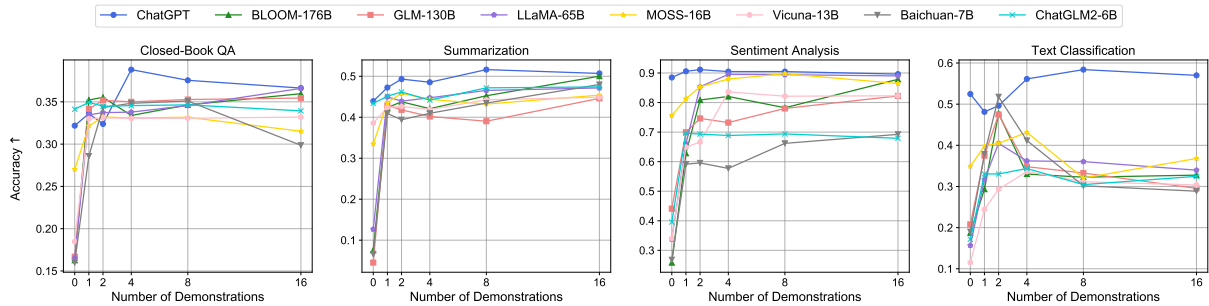


Figure 13: The accuracy of different models with various numbers of few-shot demonstrations.

it is more friendly to models without instruction tuning as they perform much better than Joint. This is because Separate restricts the model to output choices only, reducing the errors caused by unconstrained generation. However, for instruction-following models, Joint yield more advantages (e.g., ChatGLM2-6B in text classification, reading comprehension, and sentiment analysis) as some Separate prompt templates may not include all possible choices in the prompt. In this case, models are likely to produce other viable answers that could not be parsed by automatic metrics (e.g., paraphrasing the correct answer).

Similarly, we discuss the impact of Completion and Chatbot style few-shot prompting strategies, where the former concatenates everything into a string and the latter organizes demonstrations into a structured dialogue history. Figure 12 illustrates the impact of these two styles of few-shot prompting strategies in various chatbots. We see that almost all chatbots perform better with Chatbot than with Completion, demonstrating the effectiveness of this tailored strategy. We also notice that GPT-4 and ChatGPT from OpenAI are not sensitive to the few-shot prompting styles. After taking a closer look at the generation results, we find that most chatbots do not follow the format described in the instruction and illustrated in the in-context examples to customize their answers, resulting in invalid postprocessing of automatic metrics. For instance,

most prompts ask the model to output the answer only, but Claude and ChatGLM6-2B tend to provide an explanation first.

We also investigate how the performance varies as the number of in-context examples increases for Chinese LLMs. Figure 13 visualizes the overall trends of different models in different tasks. In general, most models perform better with more demonstrations and are saturated with around 4-8 training samples. In line with existing work (Liang et al., 2022), models without instruction tuning benefit more from few-shot demonstrations. We observe that many models suffer from performance degradation in the text classification task. We believe this is because our test set has a relatively large label space and including more demonstrations distracts the models.

# DOPA METER — A Tool Suite for Metrical Document Profiling and Aggregation

Christina Lohr<sup>1,2,3</sup> & Udo Hahn<sup>1,3</sup>

<sup>1</sup>Jena University Language & Information Engineering (JULIE) Lab,  
Friedrich Schiller University Jena, Germany

<sup>2</sup>Institute for Medical Informatics, Statistics and Epidemiology (IMISE)  
University Leipzig, Germany

<sup>3</sup>SMITH Consortium of the German Medical Informatics Initiative

## Abstract

We present DOPA METER, a tool suite for the metrical investigation of written language, that provides diagnostic means for its division into discourse categories, such as registers, genres, and style. The quantitative basis of our system are 120 metrics covering a wide range of lexical, syntactic, and semantic features relevant for language profiling. The scores can be summarized, compared, and aggregated using visualization tools that can be tailored according to the users' needs. We also showcase an application scenario for DOPA METER.

## 1 Introduction

The way how we encode contents in natural language utterances gives rise to linguistic divisions into registers, genres, style levels, etc. (for a thorough distinction of these terms, see Lee (2001); Biber and Conrad (2019)) that follow functional communication requirements, e.g., ease of comprehension or adherence to the wording of social peer groups. The behavioral traits indicating such divisions are manifold and range from simple token frequencies, lexical choice options (synonyms, more specific vs. more general or sublanguage vs. layman terms), via syntactic variations (easy vs. complex sentence constructions) over to pragmatic distinctions (e.g., formal vs. informal language use). Many of NLP's most pressing applied research questions (e.g., hate and fake detection, communication biases relating to people's political, religious, racial, personal orientation) are considered to be flagged this way (Xiao et al., 2022).

In this paper, we address a large variety of such behavioral aspects of language use from a metrical perspective. None of these metrics is new, but their assembly and broad coverage in a coherent tool suite and modular software framework is. We also provide means for summarization, comparison and aggregation of results and their proper visualization.

## 2 Related Work

The tool-based computational analysis of behavioral traits of language use can be divided into three branches of research: (1) *readability* checkers with language complexity measures incorporating mostly surface-level syntactic and lexico-semantic features of utterances, (2) *stylometrics* tools with strong emphasis on powerful lexico-statistical metrics, and (3) *psychometrics* devices with mostly simple frequency-based computations complemented by dictionaries with psychologically typed lexical categories.

From the perspective of *readability* (for a survey, see Collins-Thompson (2014)), the DELITE system (vor der Brück et al., 2008) can be considered as one of the language profiling systems closest to the design goals and feature types of our system. Still, its main goal, as a readability checker, is much narrower than ours. DELITE identifies and highlights passages of text which are difficult to understand (together with reasons why this is the case). To reach this goal, DELITE comes with a wide range of shallow and deep features to score the readability of documents, which is also at the heart of our work. Deep features include, e.g., topological information from dependency trees for syntactic scoring (e.g., center embedding depth, phrasal fan-out ratios) and from semantic networks for semantic scoring (number of readings per lexical entry, number of propositions per sentence, semantic network connectivity). Altogether, 48 indicators for readability at the morphological, lexical, syntactic and semantic level can be calculated, averaged per document, and a global document readability score is finally computed by applying a  $k$ -nearest neighbor classifier. The system ran on German and English input data, yet has, to the best of our knowledge, never been made publicly accessible.

In the field of *stylometrics* (for a survey, see Neal et al. (2017)), STYLO (Eder et al., 2016) has

become a *de facto* standard for the quantitative study of writing style. STYLO is an R package equipped with powerful statistical analysis modules for analytics based on frequency measurements of character- and token-based n-grams (PoS n-grams etc., not supplied by default, require externally pre-processed input). STYLO comes in two flavors. Its API allows to configure a complete processing pipeline using traditional R scripting, while it also offers a rich graphical user interface (GUI) for non-technical users to run stylometric analyses and interpret their outcome without the need for elaborate programming experience.

The seamless integration of various analytical tools under a common programming framework (making use of R's core library but also extending it by various clustering algorithms and machine learning classifiers) and its public accessibility on GITHUB<sup>1</sup> make STYLO a landmark development for stylometric tooling. Yet, STYLO does not integrate any deeper lexical, syntactic and semantic processing going beyond textual surface computations (such as distance metrics, e.g., Burrows's  $\Delta$ , very popular in the stylometric community).

The third stream of work emphasizes human lexical choice patterns in terms of the *psychometrics* of word use. Perhaps its most prominent representative is the *Linguistic Inquiry and Word Count* approach and its associated LIWC engine (Tausczik and Pennebaker, 2010).<sup>2</sup> LIWC's focus is on a categorically stratified dictionary resource (the current master dictionary comprises 6,400 words, word stems, and selected emoticons) with simple descriptive statistical tools though. LIWC reads documents word-by-word, matches each word with its dictionaries and outputs simple frequency-based lexical and PoS statistics. Overall more than 80 psychologically relevant categories ranging from linguistic ones (such as function vs. content words, parts of speech, tense markers) to psychological ones (such as Cognitive, Perceptual, and Biological Processes) are attached to single lexical entries and counted during text analysis.

LIWC was recently compared and outperformed by the SEANCE system (Crossley et al., 2017) which makes use of a range of newer, even more specialized dictionaries with a larger number of

more expressive psychological categories and variables and a higher coverage of entries. Crossley et al. (2019) use a battery of independent systems for their experiments, each one highly specialized for computing different dimensions of readability, such as syntactic complexity (177 indices from the TAASSC system (Kyle and Crossley, 2018)), lexico-semantic frequency and richness (135 indices from the TAALES system (Kyle and Crossley, 2015)), text cohesion (over 150 indices from the TAACO system (Crossley et al., 2016)), and sentiment and social cognition scores (20 indices from the SEANCE system (Crossley et al., 2017)). Hence, roughly 500 individual scores have to be assembled from these stand-alone systems and combined in an umbrella system for result merging. Alternatively (not used by Crossley et al. (2019), but playing a prominent role in many recent readability studies), COH-METRIX<sup>3</sup> (Graesser et al., 2011) provides a multi-dimensional set of (psycho)linguistic and discourse features (version 3.0 incorporates 108 different indices).

Recently, Štajner et al. (2020) introduced CoCo, an advanced system with cognitively plausible features, yet its focus is limited on conceptual complexity computation of texts. SENTSPACE (Tuckute et al., 2022) is a sentence-focused analysis engine rather close to the design goals of our work, which also uses a range of cognitively plausible lexical, syntactic and semantic features. However, it lacks classical stylometric and readability indices and is limited to analyses up to the single document level only. In contrast to this work, we aim at cross-document and cross-corpus analyses for more powerful register, genre and style analyses.

Despite the remarkable progress that has been made already—the proliferation of surface-level, linguistic and cognitive features under scrutiny, and the growing number of metrics making use of them—we observe a fundamental lack of integration of and abstraction from single counts and scores in these precursors. Accordingly, a major goal of our work is to provide reasonable summarization, comparison, and aggregation levels for single metrics so that divisions into registers, genres and styles can be computed on the fly based on the contributions of a wide range of linguistic layers (integrating lexical, syntactic, and semantic features) for complex collections of (multilingual) linguistic data in terms of (sets of) corpora.

<sup>1</sup><https://github.com/computationalstylistics/stylo>

<sup>2</sup>The most recent version, LIWC2015, is available under <http://liwc.wpengine.com/> and must be purchased for a modest fee for academic and industrial use.

<sup>3</sup><http://www.cohmetrix.com>

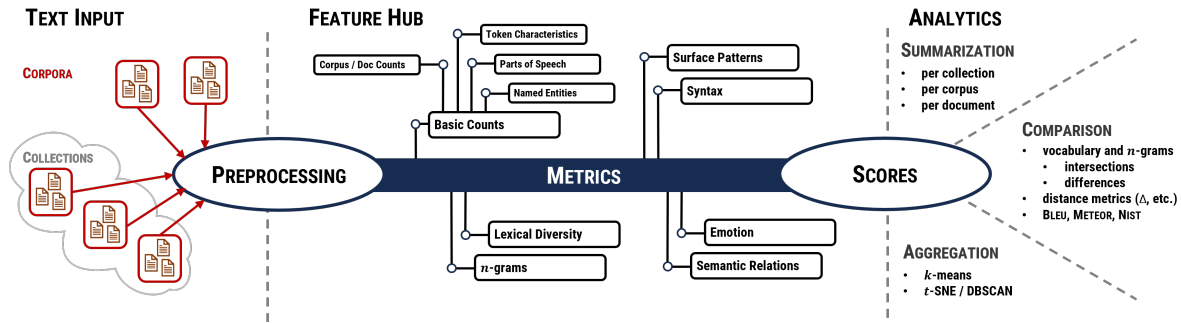


Figure 1: Overview of the building blocks of DOPA METER

### 3 DOPA METER’s System Architecture

DOPA METER is based on PYTHON and SPACY<sup>4</sup> and supports all SPACY compatible language modules. Our system is publicly accessible via GITHUB.<sup>5</sup> It is based on strict software engineering principles, such as modularity, easy resource maintenance and (re-)configuration (selection and augmentation of metrics and language resources, such as corpora and lexicons).

The three-layered architecture of DOPA METER is depicted in Fig. 1. It consists of

- arbitrarily many *text corpora* that can also be grouped into *collections of corpora* which serve as textual input channel (including a pre-processing pipeline),
- the *feature hub* that elicits relevant features from the corpora for use by a large variety of *metrics*,
- and three *analytics* layers—apart from simple report generation (summaries of metrics-derived scores), we offer a comparison mode across documents and corpora, as well as cluster-based aggregation of results.

#### 3.1 Input and Pre-processing

The input for DOPA METER consists of a set of *text corpora* that can be bundled into *collections*, for convenience. Each corpus consists of single text files, the *documents*, each of which will automatically be pre-processed and split into *sentences* and *tokens*.

#### 3.2 Feature Hub

The computation of features is divided into (1) simple *feature counts* whose results feed (2) a collection of *metrics*. We here distinguish micro statistics (at the document level) and macro statistics (at the corpus level).

The feature hub comprises sets of single features and groups them for better comprehensibility (see the discussion below and Table 3 in the Appendix). The computation of features allows for a *tailored* mode (configured by the user via choice options) or a *default* mode that takes all features into account.

##### 3.2.1 Basic Counts

In order to get started we perform basic counts of sentences, tokens, types (vocabulary size), lemmata and characters using SPACY tooling (*Corpus/Doc Counts* in Fig. 1).<sup>6</sup>

In addition, *Token Characteristics*<sup>7</sup> comprise information about alphanumeric strings, lower/upper casing, etc. The counts of *Parts of Speech* (PoS) and *Named Entities* and their tagging are derived from SPACY’s embedded language models and supply linguistically more informed feature sets.

##### 3.2.2 n-grams

*n-grams* are sequential series of (configurable)  $n=\{1,2,3,\dots\}$  tokens or (PoS) tags. The scores calculate the ratios of *n-grams* for single documents and whole corpora or corpus collections.

##### 3.2.3 Lexical Diversity

*Lexical Diversity* subsumes a group of 24 features borrowing from stylometric vocabulary metrics. Among others, this includes the common *type-token ratio (TTR)*, but also more sophisticated metrics such as *Guiraud’s R* or *Herdan’s C*. We also incorporate metrics which address the frequency spectrum of lexical items (e.g., *Sichel’s S*) and ones capturing lexical distributions over the whole document (e.g., *Moving-Average TTR*). Last but not least, we also provide metrics for *lexical density* such as the ratio of function words. For surveys of metrics of lexical diversity, see Malvern et al. (2004); Evert et al. (2017).

<sup>4</sup><https://spacy.io>

<sup>5</sup><https://github.com/dopameter/dopameter>

<sup>6</sup><https://spacy.io/usage/linguistic-features>

<sup>7</sup><https://spacy.io/api/token>



### 3.2.4 Surface Patterns

*Surface pattern* metrics, also known as *Readability* scores, mainly focus on syllable counts, token and sentence length and thus target surface-level phenomena only. Among the large number of possible choices, we included into DOPA METER 19 metrics, among them *Flesch-Kincaid*, *Dale-Chall* (for English, only), *SMOG*, *Gunning fog*, and the four *Wiener Sachtext formulas* (Bamberger and Vanacek, 1984) (for German, only). This feature class also contains a simple *Formality* score using PoS tags (Heylighen and Dewaele, 1999).

### 3.2.5 Syntax

Syntax-focused metrics account for the two major syntax representation formats: *dependency* and *constituency*. For dependency parsing, we exploit the transition-based dependency parser embedded in SPACY (Honnibal and Johnson, 2015), for constituency parsing we use the Berkeley Neural Parser (Kitaev and Klein, 2018; Kitaev et al., 2019).

The *parse metrics* take general parse graph properties into account, such as the *average maximum depth* for each parse tree, i.e., the longest path from the root node to a leaf node, the *maximum fan-out* of each parse tree, i.e., the largest number of child nodes of a node in the entire parse tree, and the inverse *average out-degree centrality* value, i.e., the number of out-going edges, computed over all dependency graphs of all sentences of a document.

### 3.2.6 Semantic Relations

We here focus on lexico-semantic resources that provide a linkage between lemmas in terms of various semantic relations. Lexicons structured this way can be regarded as semantic networks. Our focus is on relations typically provided by WORDNET-style specifications which feature synonymy, antonymy, taxonomy (hyponyms/hypernyms), and paronymy (parts and wholes).

Based on such knowledge-“heavy” resources we define several metrics that exploit the topological structures spanned in these semantic networks as instantiated by the lexical items we identify as lemmas of these lexicons within each sentence. Accordingly, we defined metrics which focus on *relational depth* by determining the minimal path length of each reading of each lemma within a document (i.e., the distance from the *top* node of the semantic network to the lemma) following taxonomic links (hypernymy or hyponymy links, only), sum up these individual length scores and average

over the number of all the lemmas’ readings, and on *semantic richness*, i.e., for each (reading of the) lemma in a sentence, we determine all semantic relation instances (i.e., hypernyms, hyponyms, parts (is-part) and wholes (has-part), antonyms) it shares with other lemmas in the lexicon and average this number over all readings per lemma in the document. Scores and their averages are also available for each individual semantic relation only (e.g., the number of hyponyms of all instantiated lemmas).

### 3.2.7 Emotion

DOPA METER supports scores for the eight fundamental emotional variables (valence, arousal, dominance, joy, anger, sadness, fear and disgust) based on dictionary look-ups incorporating the emotion lexicons from Buechel et al. (2020) in the JEMAS pipeline (Buechel and Hahn, 2016).<sup>8</sup>

## 3.3 DOPA METER’s Analytics

### 3.3.1 Summarization Mode

In the summarization mode, statistical reports of the resulting scores are generated per document and corpus (collection), including common information, such as min/max values, means, quartiles, etc. This reporting mode describes fundamental quantitative characteristics in the feature hub and can already pinpoint at differences between documents and corpora that can be deeper explored by larger-scale clustering or classification algorithms.

### 3.3.2 Comparison Mode

The comparison mode points out differences or similarities between complete text corpora or user-defined subsets therefrom. It is based on a differential analysis of the corpus vocabulary, *n*-grams and the metrics targeting different levels of linguistic analysis mentioned above.

Besides the metrics already introduced, we also make use of well-known distance metrics from the field of stylometrics and authorship detection, e.g., *Burrows’ Δ* (Burrows, 2002).

In addition to these stylometric computations, we incorporate scores originating from the field of machine translation, such as BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and NIST (Doddington, 2002).

### 3.3.3 Aggregation Mode

Going beyond the micro statistics at the single document and corpus level, the aggregation mode is

<sup>8</sup><https://github.com/JULIELab/JEmAS/releases>

able to compute dependencies between different (sets of) corpora at the macro level of analysis. With varying configurations of features,  $k$ -means and  $t$ -distributed Stochastic Neighbor Embedding ( $t$ -SNE) (van der Maaten and Hinton, 2008) with DBSCAN (Ester et al., 1996) are used as clustering algorithms at the moment. Our modular architecture, however, is open to extension by a wider range of additional clustering algorithms and other machine learning libraries.

#### 4 DOPA METER in Action

We now illustrate facets of the rich functionality of DOPA METER. Our scenario features two languages, English and German, and a broad application domain (medicine) with six corpora (collections) from a wide range of genres (see Table 1):<sup>9</sup>

| Corpus    | Documents | Sentences | Tokens    |
|-----------|-----------|-----------|-----------|
| de.Clin   | 3 497     | 145 870   | 1 649 156 |
| de.PubMed | 1 028     | 5 676     | 101 173   |
| de.SocMed | 4 000     | 30 943    | 433 999   |
| de.Wiki   | 4 400     | 326 721   | 4 348 255 |
| en.Clin   | 5 918     | 437 598   | 7 065 887 |
| en.SocMed | 3 601     | 13 168    | 172 927   |

Table 1: Quantitative data of the demo corpus collection

**de.Clin** is composed of several publicly available German clinical corpora: JSYNCC (Lohr et al., 2018), ASSESS (Miñarro Giménez et al., 2019), BRONCO (Kittner et al., 2021), GRASCCO (Modersohn et al., 2022), EX4CDS (Roller et al., 2022), CARDIO:DE (Richter-Pechanski et al., 2023) and a set of X-ray reports (Dewald et al., 2023),

**de.PubMed** contains the German subset of PUBMED abstracts featuring clinical cases,<sup>10</sup>

**de.SocMed** contains medical layman and expert expressions from a patient forum (Seiffe et al., 2020),

**de.Wiki** collects medical articles from Wikipedia including info-box data with an ICD-10 code,<sup>11</sup>

**en.Clin** incorporates public corpora supplied for the I2B2 and N2C2 challenge series,<sup>12</sup> and

**en.SocMed** combines English language TWITTER corpora with biomedical content: BEAR (Wührl and Klinger, 2022), COVERT (Mohr et al., 2022), and BIOCLAIM (Wührl and Klinger, 2021).

<sup>9</sup>Instructions how to build the corpora in order to reproduce our experiments can be found under <https://doi.org/10.5281/zenodo.10000771>

<sup>10</sup><https://pubmed.ncbi.nlm.nih.gov/>, running the query "Case Reports[Publication Type] AND GER[LA]"

<sup>11</sup><https://www.wikipedia.de/>

<sup>12</sup><https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/>

#### Summarization Mode:

The boxplots from Figure 2 depict the results from surface-level formality scoring (based on Heylighen and Dewaele (1999)) in a visual way. Clinical documents, for both languages, are in the high end of formal language use, whereas social media language, not surprisingly, scores at the lower end, with news, WIKIPEDIA, and PUBMED in between.

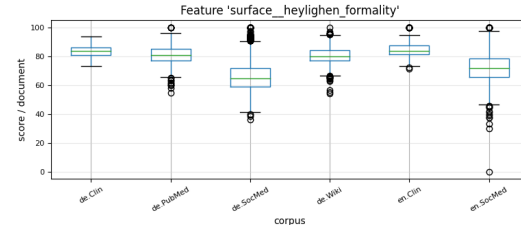


Figure 2: Surface Heylighen formality scores

Table 2 contains scores that illustrate corpus-based metrics from *Surface Patterns* (Flesch Reading Ease index), *Syntax* (depth of dependency parse trees (Dep-Depth)), and WORDNET-based *Semantic Relations* (semantic richness of synonyms).

|           | Surface | Syntax    | Semantics    |
|-----------|---------|-----------|--------------|
| Corpus    | Flesch  | Dep-Depth | Synonym-Rich |
| de.Clin   | 69.97   | 4.28      | 2.05         |
| de.PubMed | 59.91   | 4.75      | 3.45         |
| de.SocMed | 35.88   | 6.34      | 4.09         |
| de.Wiki   | 87.68   | 4.74      | 3.10         |
| en.Clin   | 85.59   | 4.98      | 0.80         |
| en.SocMed | 85.07   | 4.14      | 0.81         |

Table 2: Scores for *Flesch Reading Ease* (Flesch), average maximum depth of dependency trees (Dep-Depth), and semantic richness of synonyms from WORDNET (Synonym-Rich) (maxima in red, minima in blue)

Surprisingly, German WIKIPEDIA texts are the hardest to understand, in a similar readability range with English clinical documents and social media chats. The German expert-layman data is by far the easiest to read. German clinical documents exhibit a higher readability than English ones.

The highest syntactic complexity in terms of parse tree depth is attributed to the German expert-layman corpus (expert statements seem to suffer from ‘hard’ syntax), with no substantial differences for the remaining corpora.

The German social media corpus (in contrast to the English one) is the richest in terms of synonyms, whereas both clinical corpora are semantically poor at that level (adhering to canonical medical terminology—the English one being even poorer than the German one). The medical German WIKIPEDIA is in a similar range with German clinical and PUBMED documents on that dimension.

### Comparison Mode:

To highlight the lexical intersection among corpora, the heatmap in Fig. 3 is provided for 1-grams. The language division is obvious, yet the status of the German (medical) WIKIPEDIA is interesting insofar as it has a rather strong overlap with German PUBMED and expert-layman social media data. Furthermore, German clinical reports share a remarkable portion of vocabulary with German PUBMED and, to a lesser degree though, with expert-layman interaction in social media.

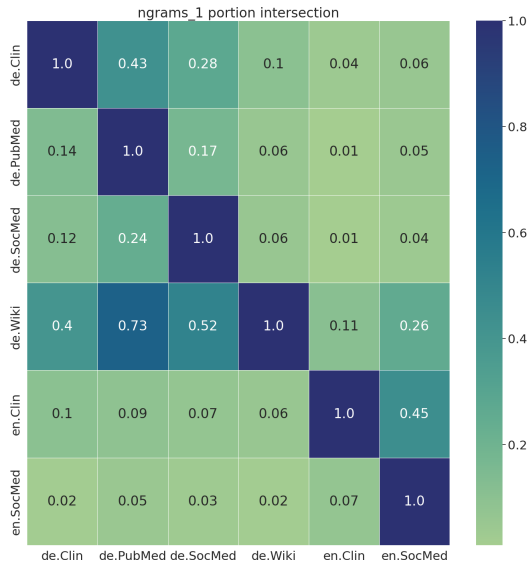


Figure 3: Vocabulary Intersection

### Aggregation Mode:

Figure 4 depicts the distribution of the scores for formal token attributes, e.g., whether a token is alphanumeric or a punctuation mark, using T-SNE (van der Maaten and Hinton, 2008), thus mapping high-dimensional data onto two dimensions.

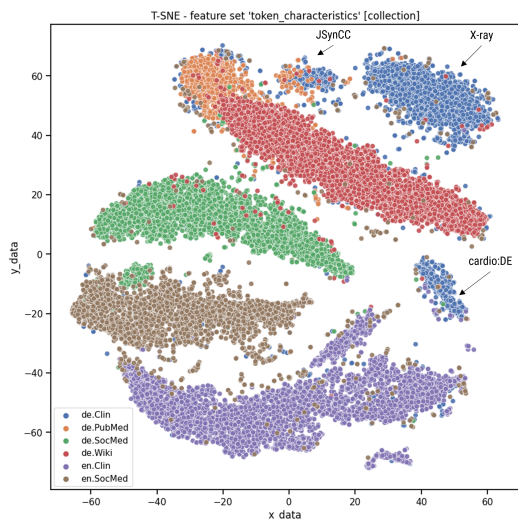


Figure 4: Clustering by token characteristics (1)

Again, the division between languages is obvious. There are clear differences between Ger-

man (upper part of Fig. 4) and English language (lower part). Social media corpora (de.SocMed and en.SocMed) of both languages lie close to each other (green and brown area) as are the samples from PUBMED and WIKIPEDIA (orange and green parts). Yet, the samples of German clinical language are divided into three distinct clusters (blue dots, with labels for the three largest corpora; for more details, see Section D in the Appendix), parts of which are close to WIKIPEDIA and PUBMED, or even overlap with those from the English language.

All these observations indicate that none of the features in isolation is capable of properly predicting specific discourse categories, such as registers or text genres. Hence, a deeper exploration of dependencies between the features we measure seems more appropriate and DOPA METER might be a suitable toolkit for this endeavor.

## 5 Conclusions

We introduced DOPA METER, a toolkit for quantifying feature distributions at the lexical, syntactic and semantic dimension. We supply 120 metrics for scoring linguistic behavior at these axes. Scores can be summarized, compared, and aggregated using flexibly tailorable visualization tools.

DOPA METER’s feature collection reflects one main design goal of our work, namely the integration of as many linguistic levels as possible, thus moving away from much more selective approaches in stylometrics and psychometrics. A second unique feature of our approach is its focus on lucid system architecture for flexible system engineering, i.e., easy maintainability and augmentation by new metrics and language resources (corpora, lexicons) in a coherent *all-in-one* system design. This contrasts with the proliferation of stylometric extensions spread over lots of local GITHUB links lacking further integration, on the one hand, and frozen system packages in the psychometric domain, on the other hand. The source code and its documentation are provided under the open MIT licence and our tool can be conveniently expanded and adapted to specific needs.

This way, DOPA METER may be useful as a metadata generator for documents and text corpora, with facilities for quantitative data description (scoring), comparison and aggregation. Such an approach may also pave the way towards an empirically sound way of routinely running NLP *data diagnostics* (Xiao et al., 2022).

## References

- Richard Bamberger and Erich Vanacek. 1984. *Lesen – Verstehen – Lernen – Schreiben*. Diesterweg.
- Douglas Biber and Susan Conrad. 2019. *Register, Genre, and Style*, 2nd edition. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK.
- Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem: dimensional models and their implications on emotion representation and metrical evaluation. In *ECAI 2016 — Proceedings of the 22nd European Conference on Artificial Intelligence. Including PAIS 2016 — Prestigious Applications of Artificial Intelligence.*, volume 2: Long Papers, pages 1114–1122, The Hague, The Netherlands, August 29 - September 2, 2016. IOS Press.
- Sven Buechel, Susanna Rücker, and Udo Hahn. 2020. Learning and evaluating emotion lexicons for 91 languages. In *ACL 2020 — Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.*, pages 1202–1217, [Seattle, Washington, USA,] July 5-10, 2020 (Virtual Event). Association for Computational Linguistics (ACL).
- John Burrows. 2002. ‘Delta’ : a measure of stylistic difference and a guide to likely authorship. *Literary and Linguistic Computing*, 17(3):267–287.
- Kevyn Collins-Thompson. 2014. Computational assessment of text readability: a survey of current and future research. *International Journal of Applied Linguistics*, 165(2):97–135.
- Scott A. Crossley, Kristopher Kyle, and Danielle S. McNamara. 2016. The tool for the automatic analysis of text cohesion (TAACO): automatic assessment of local, global, and text cohesion. *Behavior Research Methods*, 48(4):1227–1237.
- Scott A. Crossley, Kristopher Kyle, and Danielle S. McNamara. 2017. Sentiment analysis and social cognition engine (SEANCE) : an automatic tool for sentiment, social cognition, and social-order analysis. *Behavior Research Methods*, 49(3):803–821.
- Scott A. Crossley, Stephen Skalicky, and Mihai Dascalu. 2019. Moving beyond classic readability formulas: new methods and new models. *Journal of Research in Reading*, 42(3-4):541–561.
- Michael Denkowski and Alon Lavie. 2014. METEOR UNIVERSAL : language specific translation evaluation for any target language. In *WMT 2014 — Proceedings of the 9th Workshop on Statistical Machine Translation @ ACL 2014.*, pages 376–380, Baltimore, Maryland, USA, June 26-27, 2014. Association for Computational Linguistics (ACL).
- Cornelia L. A. Dewald, Alina Balandis, Lena S. Becker, Jan B. Hinrichs, Christian von Falck, Frank K. Wacker, Hans Laser, Svetlana Gerbel, Hinrich B. Winther, and Johanna Apfel-Starke. 2023. Automated classification of free-text radiology reports: using different feature extraction methods to identify fractures of the distal fibula. *RöFo — Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren*, 195(8):713–719.
- George R. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *HLT 2002 — Human Language Technology Conference. Proceedings of the 2nd International Conference on Human Language Technology Research.*, pages 138–145, San Diego, California, USA, March 24-27, 2002. Morgan Kaufmann Publishers Inc.
- Maciej Eder, Jan Rybicki, and Mike Kestemont. 2016. *Stylometry with R : a package for computational text analysis*. *R Journal*, 16(1):107–121.
- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD '96 — Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.*, pages 226–231, Portland, Oregon, USA, August 2-4, 1996. AAAI Press.
- Stefan Evert, Peter Uhrig, Sabine Bartsch, and Thomas Proisl. 2017. E-VIEW-ALATION : a large-scale evaluation study of association measures for collocation identification. In *Electronic Lexicography in the 21st Century. eLex 2017 — Proceedings of the 5th Conference on Electronic Lexicography.*, pages 531–549, Leiden, Netherlands, 19-21 September 2017. Lexical Computing CZ s.r.o.
- Arthur C. Graesser, Danielle S. McNamara, and Jonna M. Kulikowich. 2011. COH-METRIX : providing multilevel analyses of text characteristics. *Educational Researcher*, 40(5):223–234.
- Francis Heylighen and Jean-Marc Dewaele. 1999. *Formality of language: definition, measurement and behavioral determinants*. Technical report, Center "Leo Apostel", Free University of Brussels.
- Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *EMNLP 2015 — Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*, pages 1373–1378, Lisbon, Portugal, 17-21 September 2015. Association for Computational Linguistics (ACL).
- Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *ACL 2019 — Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.*, pages 3499–3505, Florence, Italy, July 28 - August 2, 2019.
- Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL 2018 — Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.*, volume 1:

- Long Papers, pages 2676–2686, Melbourne, Victoria, Australia, July 15–20, 2018. Association for Computational Linguistics (ACL).
- Madeleine Kittner, Mario Lamping, Damian T Rieke, Julian Götze, Bariya Bajwa, Ivan Jelas, Gina Rüter, Hanjo Hautow, Mario Sängler, Maryam Habibi, Marit Zettwitz, Till de Bortoli, Leonie Ostermann, Jurica Ševa, Johannes Starlinger, Oliver Kohlbacher, Nisar P Malek, Ulrich Keilholz, and Ulf Leser. 2021. [Annotation and initial evaluation of a large annotated German oncological corpus](#). *JAMIA Open*, 4(2):o0ab025.
- Kristopher Kyle and Scott A. Crossley. 2015. Automatically assessing lexical sophistication: indices, tools, findings, and application. *TESOL Quarterly*, 49(4):757–786.
- Kristopher Kyle and Scott A. Crossley. 2018. Measuring syntactic complexity in L2 writing using fine-grained clausal and phrasal indices. *The Modern Language Journal*, 102(2):333–349.
- David Y. W. Lee. 2001. Genres, registers, text types, domains and styles: clarifying the concepts and navigating a path through the BNC jungle. *Language Learning & Technology*, 5(3):37–72.
- Christina Lohr, Sven Buechel, and Udo Hahn. 2018. Sharing copies of synthetic clinical corpora without physical distribution: a case study to get around IPRs and privacy constraints featuring the German JSYNCC corpus. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation.*, pages 1259–1266, Miyazaki, Japan, May 7–12, 2018. European Language Resources Association (ELRA).
- David Malvern, Brian Richards, Ngoni Chipere, and Pilar Durán. 2004. *Lexical Diversity and Language Development: Quantification and Assessment*. Palgrave Macmillan.
- Jose A. Miñarro Giménez, Ronald Cornet, Marie Christine Jaulent, Heike Dewenter, Sylvia Thun, Kirstine Rosenbeck G, Daniel Karlsson, and Stefan Schulz. 2019. Quantitative analysis of manual annotation of clinical text samples. *International Journal of Medical Informatics*, 123:37–48.
- Luise Modersohn, Stefan Schulz, Christina Lohr, and Udo Hahn. 2022. GRASCCO : the first publicly shareable, multiply-alienated German clinical text corpus. In *German Medical Data Sciences 2022 — Future Medicine: More Precise, More Integrative, More Sustainable! Proceedings of the Joint Conference of the 67th Annual Meeting of the GMDS & 14th Annual Meeting of the TMF*, number 296 in Studies in Health Technology and Informatics, pages 66–72, [Kiel, Germany,] 21–25 August 2022 (Virtual Event). IOS Press.
- Isabelle Mohr, Amelie Wüthrl, and Roman Klinger. 2022. COVERT : a corpus of fact-checked biomedical COVID-19 tweets. In *LREC 2022 — Proceedings of the 13th International Conference on Language Resources and Evaluation.*, pages 244–257, Marseille, France, June 20–25, 2022. European Language Resources Association (ELRA).
- Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard. 2017. Surveying stylometry techniques and applications. *ACM Computing Surveys*, 50(6):#86.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU : a method for automatic evaluation of machine translation. In *ACL ’02 — Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.*, pages 311–318, Philadelphia, Pennsylvania, USA, July 6–12, 2002. Association for Computational Linguistics (ACL).
- Phillip Richter-Pechanski, Philipp Wiesenbach, Dominic M. Schwab, Christina Kiriakou, Mingyang He, Michael M. Allers, Anna S. Tiefenbacher, Nicola Kunz, Anna Martynova, Noemie Spiller, Julian Mierisch, Florian Borchert, Charlotte Schwind, Norbert Frey, Christoph Dieterich, and Nicolas A. Geis. 2023. A distributable German clinical corpus containing cardiovascular clinical routine doctor’s letters. *Scientific Data*, 10:#207.
- Roland Roller, Aljoscha Burchardt, Nils Feldhus, Laura Seiffe, Klemens Budde, Simon Ronicke, and Bilgin Osmanodja. 2022. An annotated corpus of textual explanations for clinical decision support. In *LREC 2022 — Proceedings of the 13th International Conference on Language Resources and Evaluation.*, pages 2317–2326, Marseille, France, June 20–25, 2022. European Language Resources Association (ELRA).
- Laura Seiffe, Oliver Marten, Michael Mikhailov, Sven Schmeier, Sebastian Möller, and Roland Roller. 2020. From witch’s shot to music making bones: resources for medical laymen to technical language and vice versa. In *LREC 2020 — Proceedings of the 12th International Conference on Language Resources and Evaluation.*, pages 6185–6192, Marseille, France, May 11–16, 2020. European Language Resources Association (ELRA).
- Yla R. Tausczik and James W. Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54.
- Greta Tuckute, Aalok Sathe, Mingye Wang, Harley Yoder, Cory Shain, and Evelina Fedorenko. 2022. SENTSPACE : large-scale benchmarking and evaluation of text using cognitively motivated lexical, syntactic, and semantic features. In *NAACL-HLT 2022 — Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations Session.*, pages 99–113, Seattle, Washington, USA, July 10–15, 2022 (and Virtual Event). Association for Computational Linguistics (ACL).

- Laurens van der Maaten and Geoffrey Hinton. 2008. [Visualizing data using t-SNE](#). *Journal of Machine Learning Research*, 9(86):2579–2605.
- Tim vor der Brück, Sven Hartrumpf, and Hermann Helbig. 2008. A readability checker with supervised learning using deep indicators. *Informatica*, 32(4):429–435.
- Sanja Štajner, Sergiu Nisioi, and Ioana Hulpuş. 2020. CoCo : a tool for automatically assessing conceptual complexity of texts. In *LREC 2020 — Proceedings of the 12th International Conference on Language Resources and Evaluation*, pages 7179–7186, Marseille, France, May 11-16, 2020. European Language Resources Association (ELRA).
- Amelie Wühlr and Roman Klinger. 2021. Claim detection in biomedical TWITTER posts. In *BioNLP 2021 — Proceedings of the 20th Workshop on Biomedical [Natural] Language Processing @ NAACL-HLT 2021.*, pages 131–142, June 11, 2021 (Virtual Event). Association for Computational Linguistics (ACL).
- Amelie Wühlr and Roman Klinger. 2022. Recovering patient journeys: a corpus of biomedical entities and relations on TWITTER (BEAR). In *LREC 2022 — Proceedings of the 13th International Conference on Language Resources and Evaluation.*, pages 4439–4450, Marseille, France, June 20-25, 2022. European Language Resources Association (ELRA).
- Yang Xiao, Jinlan Fu, Weizhe Yuan, Vijay Viswanathan, Zhoumianze Liu, Yixin Liu, Graham Neubig, and Pengfei Liu. 2022. DATA LAB : a platform for data analysis and intervention. In *ACL 2022 — Association for Computational Linguistics: System Demonstrations.*, pages 182–195, Dublin, Ireland, May 22-27, 2022 (and Virtual Event). Association for Computational Linguistics (ACL).

## A Ethical Considerations

DOPA METER uses a wide range of external resources, such as corpora, lexicons or terminology systems with potentially built-in biases. Users of DOPA METER should be sensitive towards potential pitfalls when analyzing data and reporting the results gathered with DOPA METER.

## B Limitations

DOPA METER combines metrics, e.g., for readability or syntactic complexity, which are commonly used but often lack comparative evaluation. Hidden, and potentially unrecognized or unwarranted, dependencies between them should be carefully considered.

Despite our efforts to include at least two languages (English and German), the multilingual dimension needs further elaboration. When doing so one might encounter shortcomings or even gaps for particular languages (e.g., for readability formulae, corpora, terminologies or lexicons).

Finally, DOPA METER’s aggregation component needs further extension by complementary clustering and ML classification algorithms.

## C Grants

This work was supported by BMBF within the SMITH project under grants 01ZZ1803G and 01ZZ1803A such as the GeMTeX project under grant 01ZZ2314D.

## D Fine-Grained Clustering of All Individual Corpora

The following figure provides a more detailed view of the data aggregated in Fig. 4.

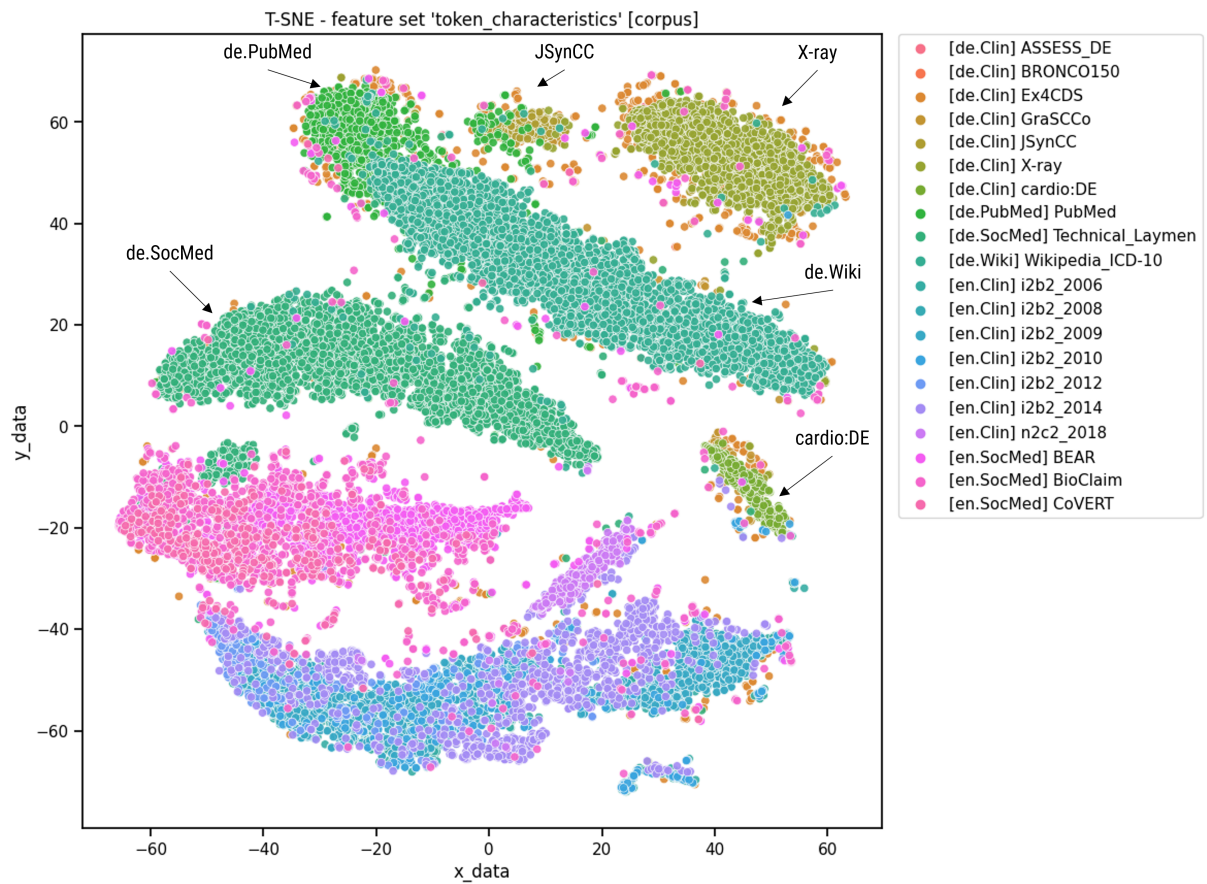


Figure 5: Clustering by token characteristics (2): Finer-grained visualization of Fig. 4

## E Feature Hub Summary

| Feature Hub                  | Metrics  | Amount of Metrics |         |       | Modus / Analysis |         |         |
|------------------------------|--|-------------------|---------|-------|------------------|---------|---------|
|                              |  | German            | English | Mult. | Count            | Metrics | Compare |
| <b>Corpus / Doc Counts</b>   | <i>characters, sentences, different_sentences, tokens, types, lemmata</i>  | 6                 | 6       | 6     | ✓                |         |         |
| <b>Token Characteristics</b> | <i>is_alpha, is_ascii, is_digit, is_lower, is_upper, is_title, is_punct, is_left_punct, is_right_punct, is_space, is_bracket, is_quote, is_currency, like_url, like_num, like_email, is_stop</i>   | 17                | 17      | 17    | ✓                | ✓       |         |
| <b>Part of Speech</b>        | depends on spaCy language model<br>German (de_core_news_sm): TIGER tagset (e.g., DET, NOUN, VERB, ADP, ...)<br>English (en_core_web_sm): Onto Notes 5 (e.g., AUX, NOUN, VERB, PROPN, ...)  | 1                 | 1       | 1     | ✓                | ✓       |         |
| <b>Named Entities</b>        | depends on spaCy language model<br>German (de_core_news_sm): WikiNER (only LOC, PERS, MISC, ORG)<br>English (en_core_web_sm): WordNet 3.0 (e.g., DATE, LOC, PERSON, ORG)   | 1                 | 1       | 1     | ✓                | ✓       |         |
| <b>n-grams (ftidf)</b>       | depends on configuration of n and most frequent words, preferred: n={1,2,3}  | 1                 | 1       | 1     | ✓                | ✓       | ✓       |
| <b>Lexical Diversity</b>     | <i>type_token_ratio, lexical_density, guiraud_r, herdan_c, dugast_k, maas_a2, dugast_u, tuldava_ln, brunet_w, ctrr, summer_s, str, sichel_s, michea_m, honore_h, entropy, yule_k, simpson_d, herdan_ym, hdd, evenness, matrr, mtlD</i>   | 23                | 23      | 23    | ✓                | ✓       | ✓       |
| <b>Surface Patterns</b>      | <i>avg_token_len_chars, avg_sent_len_tokens, avg_sent_len_chars, flesch_kincaid_grade_level, smog, coleman_liaw, ari, forcast, gunning_fog, heylighen_formality</i><br>no default: <i>toks_min_three_syllables, toks_larger_six_letters, toks_one_syllable, syllables letter_tokens no_digit_tokens</i><br>only German: <i>flesch_reading_ease, wiener_sachtextformel_1, wiener_sachtextformel_2, wiener_sachtextformel_3, wiener_sachtextformel_4</i><br>only English: <i>flesch_reading_ease, dale_chall</i> | 23                | 20      | 18    | ✓                | ✓       | ✓       |
| <b>Syntax - Dependency</b>   | AvgFan, MaxFan, AvgMaxDepth, AvgDepDist, MaxDepDist, AvgOutdegreeCentralization, AvgClosenessCentralization, occurrences of tree nodes (depending on spaCy language model)   | 8                 | 8       | 8     | ✓                | ✓       |         |
| <b>Syntax - Constituency</b> | AvgMaxDepth, AvgFan, MaxFan, AvgNonTerminales_sent, AvgConstituents_sent, AvgTunits_sent, AvgLenConstituents, AvgLenTunits, AvgOutdegreeCentralization, MaxOutdegreeCentralization, AvgClosenessCentralization, MaxClosenessCentralization occurrences of tree nodes   | 13                | 13      | 13    | ✓                | ✓       |         |
| <b>Emotion</b>               | valence, arousal, dominance, joy, anger, sadness, fear, disgust  | 8                 | 8       | 8     |                  | ✓       |         |
| <b>Semantic Relations</b>    | <i>sem_rich_hyponyms, sem_rich_hyponyms, sem_rich_taxonyms, sem_rich_antonyms, sem_rich_synonyms, sem_rich_meronyms, sem_rich_holonyms, sem_rich_min_depths_avg, min_depths_min, min_depths_max, max_depths_avg, max_depths_min, max_depths_max, synsets_avg, senses_avg</i> , occurrences of synsets, occurrences of senses   | 18                | 18      | 18    | ✓                | ✓       | ✓       |
| <b>Amount of all Metrics</b> |  | 119               | 116     | 114   |                  |         |         |

Table 3: Summary of all *Feature Hubs* and all *Metrics* of DOPA METER



# MUTED : Multilingual Targeted Offensive Speech Identification and Visualization

Christoph Tillmann, Aashka Trivedi, Sara Rosenthal, Santosh Borse  
Rong Zhang, Avirup Sil, Bishwaranjan Bhattacharjee  
IBM Research AI

## Abstract

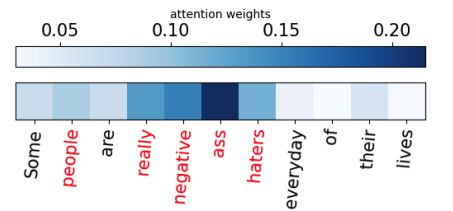
Offensive language such as hate, abuse, and profanity (HAP) occurs in various content on the web. While previous work has mostly dealt with sentence level annotations, there have been a few recent attempts to identify *offensive spans* as well. We build upon this work and introduce MUTED, a system to identify multilingual HAP content by displaying offensive ARGUMENTS and their TARGETS using heat maps to indicate their intensity. MUTED can leverage any transformer-based HAP-classification model and its attention mechanism out-of-the-box to identify toxic spans, *without further fine-tuning*. In addition, we use the spaCy library to identify the specific TARGETS and ARGUMENTS for the words predicted by the attention heatmaps. We present the model’s performance on identifying offensive spans and their targets in existing datasets and present new annotations on German text. Finally, we demonstrate our proposed visualization tool on multilingual inputs.

## 1 Introduction

Offensive language such as hate, abuse, and profanity (HAP) occurs in various content on the web such as social media sites (e.g. Twitter) and discussion forums (e.g. Reddit). Such content can be hurtful to the reader, and identifying and visualizing HAP speech is necessary to understand and avoid harm. It increases interpretability and can be used to hide and provide a warning for offensive terms, and to avoid generating hate in large language models.

While such visualizations exist, the focus has primarily been on English HAP and on identifying offensive language on the sentence level (McMillan-Major et al., 2022). There are few works that explore spans and other languages (Ranasinghe and

**WARNING:** This paper contains offensive examples.



(a) Attention Heatmap

Some people TARGET are really negative ass haters ARG everyday of their lives

(b) SpaCy: visually identifying Target and Argument

Figure 1: Example system output that shows the intensity of the offensive ARGUMENT and its TARGET, <T,A>: (a), (b): <people, really negative ass haters> .

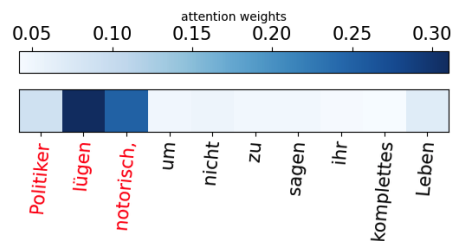


Figure 2: German Input ("Politicians notoriously lie, not to say their entire lives"): <Politiker, lügen notorisch> .

Zampieri, 2021; Wright et al., 2021) but these do not identify and visualize the TARGET of the offensive ARGUMENT which is an important indicator regarding whether the offensive argument is harmful or not, as shown in Zampieri et al. (2023).

We propose identifying hate using existing approaches (Caselli et al., 2021) to display multilingual offensive ARGUMENTS and their TARGETS using heat maps as a means of showing their intensity. Moreover, the spaCy library (Honnibal et al., 2020) can also be used to identify the specific target and argument from the predicted words. An example with a <T,A> pair is shown for English

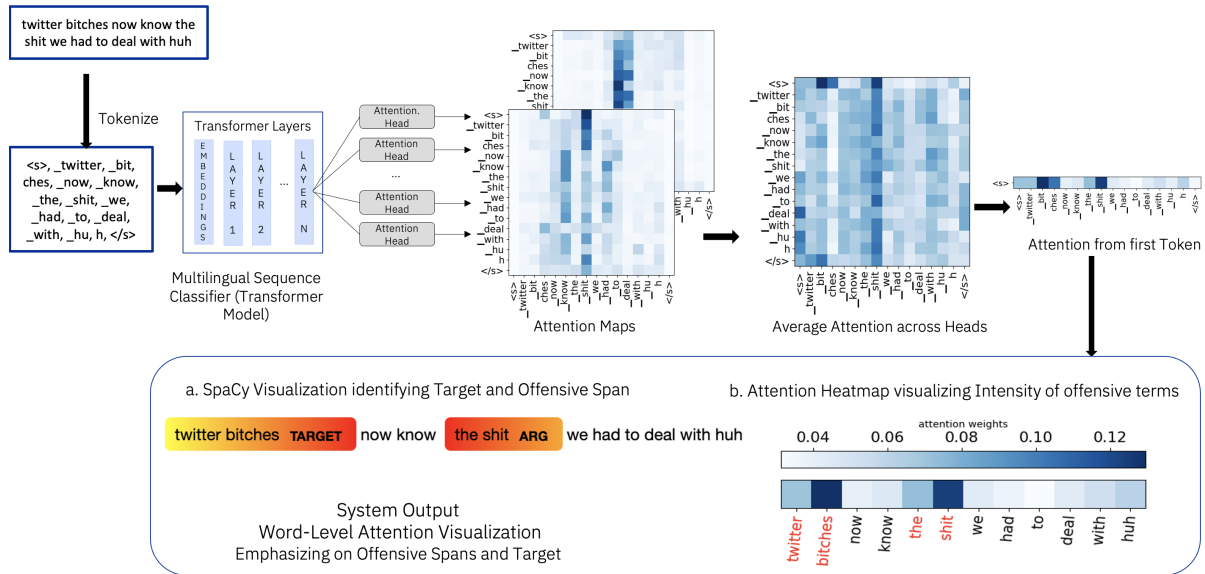



Figure 3: Muted : Visualizing offensive spans and targets using Attention Heatmaps. A token-level attention score of a given sentence is obtained using the average attention across all heads of the last layer of the given HAP classifier, and extracting the attention from the first token (often the CLS vector). The score for a word is calculated as the maximum token-level attention score of its constituent tokens. Finally, we display the predicted spans using the attention heatmap, and use spaCy’s dependency parser to identify the target and argument in the predictions.

and German inputs in Fig. 1 and Fig. 2, with the resulting visualizations. Our contributions are as follows:

- We present Muted: A **M**ultilingual **T**argeted **D**emonstration providing an intuitive way of visualizing existing classifiers by using transformer attention to identify the target of the offensive text as well as the offensive span.
- Unlike similar token classification techniques (Ranasinghe and Zampieri, 2021), our system can be used with off-the-shelf hate/abuse/profanity detectors.
- Our approach is multilingual and we demonstrate it on English (Zampieri et al., 2023) and a new German targeted offensive speech dataset. In the future, we plan to extend to more languages, e.g. a Spanish data set.
- We present easy-to-use Python notebooks and a front-end UI to run our approach on any encoder-only HAP classifier to visualize the offensive  $\langle T, A \rangle$  pair using heat-maps and spaCy <sup>1</sup>.

The rest of this paper describes related work, our approach for detecting offensive speech, and our model which outperforms existing sentence classifiers on the TBO (Zampieri et al., 2023) and TSD (Pavlopoulos et al., 2021) datasets. Finally, we present our system demonstration and its efficiency.

<sup>1</sup><https://spacy.io/api/dependencyparser>

## 2 Related Work

Identifying offensive content has been a popular area of research in recent years (Davidson et al., 2017; Jahan and Oussalah, 2023). One popular model that is available is HateBERT (Caselli et al., 2021) which is a Bert-based model finetuned on offensive speech from Reddit comments. Similar models exist in other languages such as deHateBERT (Aluru et al., 2020) in German. We present our own multilingual model for detecting offensive content which outperforms HateBERT on offensive span selection. However, our notebooks demonstrating our approach for identifying the offensive target and argument can be used with any transformer-based offensive classifier.

Several demos on offensive text exist that perform on the span or sentence level, mostly in English (McMillan-Major et al., 2022; Wright et al., 2021). Perhaps the most relevant demo is Muded (Ranasinghe and Zampieri, 2021). They identify offensive spans in input text by classifying each token as offensive or not, and support English, Danish and Greek. The UI is token-classification based, and can be used with their trained models and the datasets used in the paper (or any input text) to identify offensive spans which will be displayed in red. In contrast to other prior work, our heat map-

based system can be used to visualize the offensive argument and target for any language for which a sentence level hate classifier is available.

### 3 Approach

MUTED provides an intuitive visualization of existing HAP classifiers by using attention maps to identify offensive text and their targets, as shown in Fig. 3. Formally, for a transformer model (of  $L$  transformer layers and  $H$  attention heads) finetuned to classify whether a given input sentence  $x$  contains offensive language, we first obtain the attention outputs  $A_i^L \in \mathbb{R}^{|x| \times |x|}$ ,  $i \in [1, H]$  of the last transformer layer. We then compute the average attention across all heads,  $A' = \frac{1}{H} \sum_{i=1}^H A_i^L$ , and extract the attention vector for the first token (e.g., the CLS token for BERT (Devlin et al., 2019) models),  $A'_0 \in \mathbb{R}^{1 \times |x|}$ . Based on a threshold, we obtain the set of tokens  $T$  with the highest attention score, which can be intuitively viewed as the tokens that contribute most to the classification decision. We convert the token-level attentions into word-level attentions by assigning a word the maximum attention of any of its constituent tokens. We provide the word-level attention visualization in the form of heat maps, and mark the target and the argument of the offensive span in the sentence (see the System Output in Fig. 3).

Our system can be used to visualize any transformer-based model that is trained to classify if a given sequence has HAP content or not, irrespective of the language. In this work, we present the *Piccolo-HAP classifier*<sup>2</sup>, a tiny 4-layer XLM-Roberta (Conneau et al., 2020) model (with 153 Million parameters) finetuned on the HAP detection task for 6 languages (English, German, Japanese, Spanish, French and Portuguese). Specifically, we distil the self-attention relations of an in-house XLM-Roberta Base Model on a task-agnostic (general purpose) manner into a 4-layer architecture, as proposed in Wang et al. (2021). We finetune this general purpose language model on the HAP classification objective, using open-source multilingual annotated datasets (Founta et al., 2018; Davidson et al., 2017; Röttger et al., 2021; de Gibert et al., 2018; Ousidhoum et al., 2019; Jigsaw, 2019; Pereira-Kohatsu et al., 2019; Wiegand et al., 2018; Roß et al., 2016; Leite et al., 2020) originating from social media data, as well as internally an-

<sup>2</sup><https://medium.com/@alex.lang/fair-is-fast-and-fast-is-fair-ibm-slate-foundation-models-for-nlp-3508412a4b04>

notated samples from CC100 (Conneau et al., 2020) and scraped news data from the internet in the six languages mentioned above. For non-English data, we also translate English datasets (Davidson et al., 2017; Founta et al., 2018) to the language required. We finetune the model on a total of 1.7 million sentences, with the majority of data being in English.

## 4 Experiments

We compare our model to a random baseline, as well as open-source toxicity classifiers (monolingual and multilingual). First, we evaluate a **random** selection of spans as target and arguments in the sentence. Specifically, each span in the sentence is marked as HAP with a probability of 0.50. We also use three off-the-shelf English **HateBERT** models (Caselli et al., 2021), each finetuned on either Hateval (Basile et al., 2019), Offenseval (Zampieri et al., 2019b) or Abuseval (Caselli et al., 2020). These models were made available by the HateBERT authors<sup>3</sup>, and we have not finetuned them ourselves. We also compare our multilingual model to another open-source multilingual classifier available on HuggingFace, **Multilingual Toxicity Classifier Plus [MTC+]**<sup>4</sup>, and two German (monolingual) classifiers, **DeHateBERT-de**<sup>5</sup> (Aluru et al., 2020) and **German Toxicity Classifier Plus (V2)**<sup>6</sup>.

### 4.1 Datasets

For experiments, we use the following datasets, all of which contain data that is already known to be offensive. The data is converted into a span-selection task, where the classification model is used to identify the toxic spans (and the target of the span when applicable), using the attention maps.

- Target Based Offensive Language dataset (TBO) (Zampieri et al., 2023): TBO contains around 4500 examples of English twitter data that has been found to be offensive (Zampieri et al., 2019a; Rosenthal et al., 2021), providing token-level annotations and identifying both the offensive spans (ARGUMENT) and its TARGET in the input text. Each tweet can have multiple <T,A> pairs, and may have a "null" target if the target of the offense is not mentioned in the text. For this demonstration we did not explore the Harmful

<sup>3</sup>Model Repository for HateBERT: <https://osf.io/tbd58/>

<sup>4</sup>EIStakovskii/xlm\_roberta\_base\_multilingual\_toxicity\_classifier\_plus

<sup>5</sup>Hate-speech-CNERG/dehatebert-mono-german

<sup>6</sup>EIStakovskii/german\_toxicity\_classifier\_plus\_v2

| Model                                    | TSD: F1 Score $\uparrow$ | English TBO: F1 Score $\uparrow$ |             |             |
|--|--------------------------|----------------------------------|-------------|-------------|
|  | TARGET ONLY              | TARGET + ARG.                    | ARG. ONLY   | TARGET ONLY |
| Random                                   | 0.08                     | 0.19                             | 0.16        | 0.13        |
| HateBERT (AbusEval)                      | 0.15                     | 0.36                             | 0.30        | 0.24        |
| HateBERT (HatEval)                       | 0.16                     | 0.36                             | 0.30        | 0.27        |
| HateBERT (OffenseEval)                   | 0.23                     | 0.43                             | 0.37        | <b>0.34</b> |
| HF Multilingual Toxicity Classifier Plus | 0.29                     | 0.36                             | 0.31        | 0.22        |
| Piccolo-HAP (Ours)                       | <b>0.51</b>              | <b>0.50</b>                      | <b>0.50</b> | 0.32        |

Table 1: Results on the TSD and TBO datasets (English). Best results in bold.

| Model                                    | German TBO: F1 Score $\uparrow$ |             |             |
|--|---------------------------------|-------------|-------------|
|  | TARGET + ARG.                   | ARG. ONLY   | TARGET ONLY |
| Random                                   | 0.14                            | 0.11        | 0.08        |
| DeHateBERT (monolingual)                 | 0.17                            | 0.16        | 0.05        |
| HF German Toxicity Classifier Plus V2    | 0.19                            | 0.28        | <b>0.21</b> |
| HF Multilingual Toxicity Classifier Plus | 0.33                            | 0.23        | 0.15        |
| Piccolo-HAP (Ours)                       | <b>0.44</b>                     | <b>0.34</b> | <b>0.21</b> |

Table 2: Results on the German TBO dataset . Best results in bold.

label assigned to each tweet. We evaluate on the 475 test examples.

- German TBO: We evaluate our model on another language by annotating a small evaluation set of offensive German tweets from the GermEval corpus (Wiegand et al., 2018). Two skilled German speaking annotators were trained in the English TBO annotation task, excluding the Harmful label. In total, 255 German tweets were annotated.
- Toxic Spans Detection (TSD) (Pavlopoulos et al., 2021): The toxic spans detection task (Sem-Eval 2021 Task-5) annotated English toxic comments at the span level, marking spans of text that contribute to the offensive score. They release code that evaluates predictions at the character level.

For both TBO datasets, we experiment with using our attention-based approach to identify both the TARGET and ARGUMENT (TARGET + ARG.), only the ARGUMENT (ARG. ONLY) and only the TARGET (TARGET ONLY). In the TARGET ONLY setting, we exclude the examples that have no TARGET and only evaluate on the remaining examples; 342 English sentences, and 229 German sentences. We find this to be a fairer evaluation of our attention-based approach, as for sentences without a target the model may still produce argument spans as a prediction (as argument spans will always be attended to heavily). We leave evaluation on the NULL target examples as future work.

Note, our models are not trained on any of the above datasets, we only use them as a tool to evaluate our attention-based span detection approach for

available HAP classification models.

## 4.2 Results

We re-format each dataset as a span identification task, where the output of our system is the character-level spans for the predicted offensive arguments/targets (the spans are computed using attention maps, as described in Section 3). The F1 scores are computed on a character level, following the approach of Pavlopoulos et al. (2021). Here, the training set is used to identify the best attention thresholds to choose the offensive spans, and the test set for evaluating model performance.

Table 1 compares the results of our models to the baseline models on the English datasets. As shown, our model strongly outperforms the HateBERT classification models, the MTC+ Classifier, and the random baseline on the TSD task. We evaluate models on the the TBO dataset under three settings, and show that our models significantly outperform all baselines on identifying both the target and argument, and only the argument. On identifying only the target, it is slightly behind HateBERT finetuned on OffensEval.

The results on the German TBO dataset are shown in Table 2. We follow the same experimental setup as for the English results in Table 1 and present separate results for predicting both target and argument individually and jointly. Our Piccolo-HAP model outperforms all other German Huggingface models and also the multi-lingual model with the exception of the target-only score by the

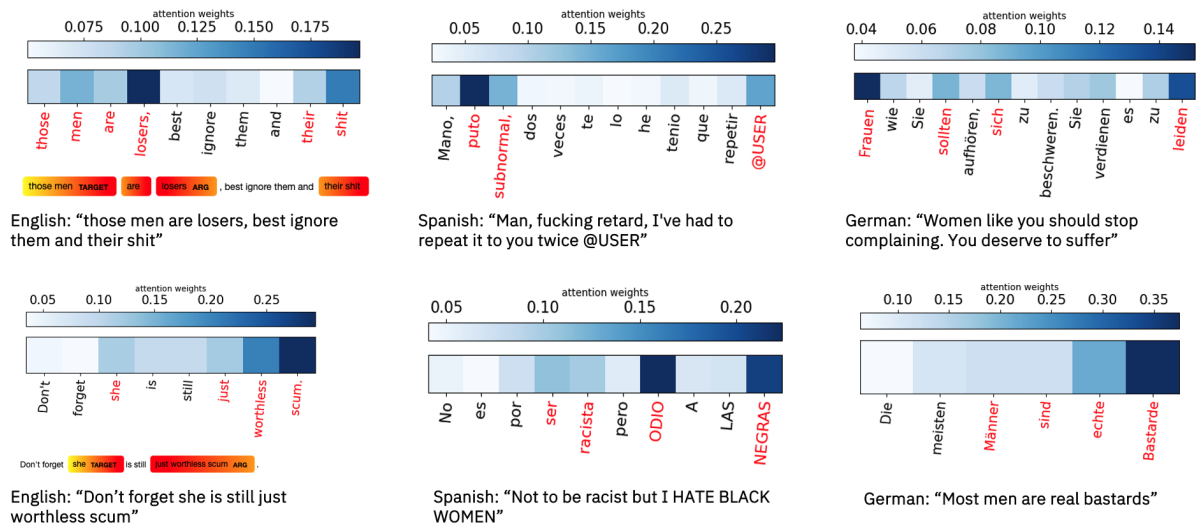


Figure 4: System outputs for examples in English, Spanish, and German. Offensive spans and targets marked in red, and images are captioned with the English translations of the input.

## HF German Toxicity Classifier.

As seen for all models, predicting both TARGET and ARGUMENT is an easier task than predicting each individually, with Target-only being the hardest setting. A way to improve the performance on this task is to modify the existing method of using the CLS token's attention to identify targets, and instead use the attention of the argument to identify the target. We leave this as future work.

For further understanding, we analyze a set of sentences from the English TBO dataset for which our model performs poorly in the Target-only setting. We find no clear patterns in this data, however we do find that our approach works very well when the targets themselves are described using offensive or derogatory terms (e.g. "these bitches", "little twats", "clowns", "idiots"). Moreover, our model does not correctly identify targets containing typos (which are common in tweets), such as *yal* instead of *y'all*. As part of future work, a spelling corrector and parser can be built into the HAP prediction system, along with current attention-based thresholds. We also analyze our model's output for some test cases where there is a NULL target annotated in the gold data, and find that our model may predict spans that could be interpreted as the TARGET. For example, the text *"The rich white people don't give a fuck about you unless you affect their bottom line"* marks NULL target in the gold data, but our model outputs *"the rich white people"* as one span, which could be interpreted as the TARGET of offensive ARGUMENT (*"don't give a fuck"*).

## 5 System Demonstration

We have Jupyter Notebooks and a front-end UI where users can load their models, and obtain visualizations for inputs in any language.

### 5.1 Jupyter Notebook

We have created a Python Jupyter notebook for displaying the <T,A> offensive pairs in a sentence. The notebook will load any encoder-only sentence level offensive classifier. It can be used on multilingual models trained on any language (e.g. English and German as we presented in our experiments). Given a sentence, we generate a heat map using the attention of the model. Then, we identify the offensive TARGETS and ARGUMENTS using a threshold on the attention. We use the *subj* and *obj* labels from the spaCy dependency parser to identify the TARGET (subject) and ARGUMENT (object) of offense. Finally, we use the spaCy visualization tools to render the sentence with the offensive TARGETS and ARGUMENTS<sup>7</sup>. Example visualizations for inputs in several languages are shown in Fig. 4. We would like to extend the tool to more languages based on multi-lingual parsing models.

### 5.2 User Interface

The MUTED user interface allows the user to play with the HAP classification model without having to know any technical details. The user interface is implemented in Flask which is a lightweight native

<sup>7</sup>In English only as the spaCy German parser did not provide the proper information to identify the target and argument.

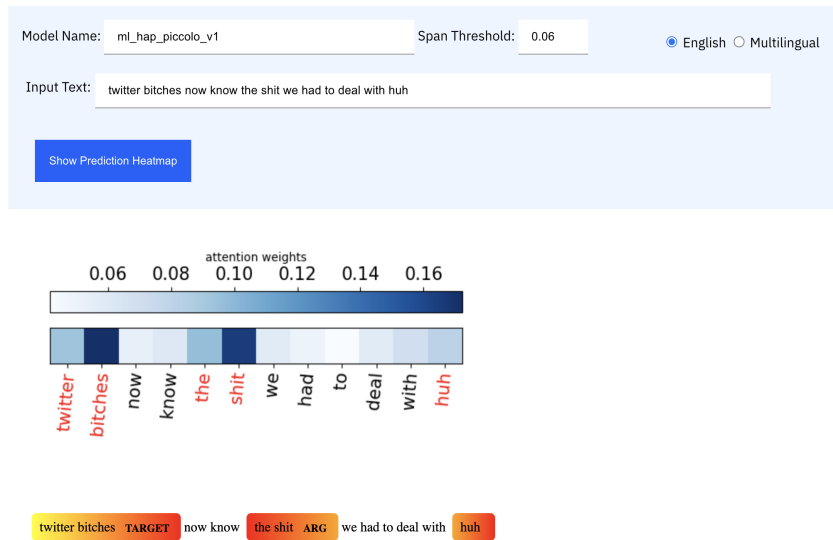


Figure 5: Screen-shot of MUTED User Interface: The user inputs the model name and input text, and selects the language and attention threshold. The system produces the attention heatmap, and (for English inputs) the spaCy visualization marking the target and argument.

python web application framework. We show an example of the user interface in Fig. 5. The UI allows the user to input the sentence, select if the language is English or non-English and select the value of Span Threshold. Upon clicking "Show prediction Heatmap", the UI renders the output visualizations on the same page. Same page rendering allows the user to tune the output with the best possible parameter values.

### 5.3 System Efficiency

We evaluate the time taken to produce the predictions and visualizations for a single input by averaging the inference time for 100 English texts. Note that the major difference between the CPU and GPU latencies is contributed by the time taken to make a prediction (which happens on the GPU when available). The visualizations always happen on the CPU, and also utilize more time.

We show the results for two multilingual models-our Piccolo HAP classifier (a 4-layer model with 153 million parameters), and the MTC+ Classifier (a 12-layer model with 277 million parameters). It takes 0.65/0.64s on CPU/GPU to run with our small model, and 0.76/0.65s on CPU/GPU for the base size model, for a single input. Table 3 shows the average latency of a single input for the different steps in the process. Thus, the system is quite efficient, and can process 100 examples in about a minute on both CPU and GPU.

|            |                 | <b>Piccolo Model</b> | <b>MTC+ Model</b> |
|------------|-----------------|----------------------|-------------------|
| <b>CPU</b> | Span Prediction | 0.02                 | 0.11              |
|            | Attention Map   | 0.22                 | 0.23              |
|            | SpaCy Visuals   | 0.41                 | 0.42              |
| <b>GPU</b> | Span Prediction | 0.01                 | 0.02              |
|            | Attention Map   | 0.22                 | 0.22              |
|            | SpaCy Visuals   | 0.41                 | 0.41              |

Table 3: Time taken (s) for span prediction and visualization of a single input. Avg. metric reported over 100 sentences, using a single core CPU and V100 GPU.

## 6 Conclusion

We present a method for identifying and visualizing offensive arguments and their targets using the attention of the sentence-based offensive classifier to create a heat map. Our multilingual model outperforms existing popular approaches on multiple datasets in English and German. We provide a notebook and user interface to run any multilingual transformer classifier on sentences and visualize the heat map as well as the <T,A> pair using spaCy visualization. In the future, we would like to add a classifier to indicate harm of the <T,A> pair as described in the TBO paper. We would also like to extend our demo to provide warnings and hide the offensive content to users.

## Ethics Statement

### Limitations

In this work, we focus primarily on English and German offensive texts. While our Piccolo model supports 6 languages, and there exists open source HAP classification models of many languages, there is a limitation on datasets available for testing. Specifically, we primarily test on datasets that include annotations of the target of offense, which are not widely available. Creating such datasets for multiple languages would be an interesting direction for future research. Moreover, the test sets that we evaluate on are relatively small in size, and consist of shorter text spans such as tweets.

As mentioned, a more robust way to use transformer attentions to identify the target of the offense is to find the words most heavily attended to by the tokens in offensive span (argument), instead of the CLS vector. In this approach, some type of aggregation strategy would be needed to select the correct tokens from the span, and we aim to implement this as part of future work.

### Intended Use

Detecting offensive content is an important task that is necessary for avoiding harm. While hateful and harmful content is used to train the models, our intended use is solely for the purpose of avoiding and removing such content and we do not support any malicious or unintended use.

### Biases

Due to the subjective nature of the task, our German annotated dataset may have unintended biases. These kind of biases are unintentional and will be prevalent in any subjective task. Anyone that uses the data should be aware that such biases may exist. Our TBO annotations are built on top of the existing GermEval dataset (Wiegand et al., 2018). We also use the TBO (Zampieri et al., 2023) and TSD(Pavlopoulos et al., 2021) dataset. Any biases in those original datasets will exist in ours as well which may impact the trained model.

### Acknowledgement

We would like to acknowledge the work of our annotators who worked on the German TBO set: Eva-Maria Wolfe, Joekie Gurski, and Mohamed Nasr.

## References

- Sai Saket Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. 2020. Deep learning models for multilingual hate speech detection. *arXiv preprint arXiv:2004.06465*.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. [SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. [HateBERT: Retraining BERT for abusive language detection in English](#). In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.
- Tommaso Caselli, Valerio Basile, Jelena Mitrović, Inga Kartoziya, and Michael Granitzer. 2020. [I feel offended, don't be abusive! implicit/explicit messages in offensive and abusive language](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6193–6202, Marseille, France. European Language Resources Association.
- 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.
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. [Automated hate speech detection and the problem of offensive language](#). *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):512–515.
- 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.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. [Large scale crowdsourcing and characterization of twitter abusive behavior](#).

- Matthew Honnibal, Ines Montani, Sofie Van Lan-deghem, and Adriane Boyd. 2020. [spacy: Industrial-strength natural language processing in python](#).
- Md Saroar Jahan and Mourad Oussalah. 2023. [A systematic review of hate speech automatic detection using natural language processing](#). *Neurocomputing*, 546:126232.
- Jigsaw. 2019. [Jigsaw unintended bias in toxicity classification data](#).
- João A. Leite, Diego F. Silva, Kalina Bontcheva, and Carolina Scarton. 2020. [Toxic language detection in social media for brazilian portuguese: New dataset and multilingual analysis](#).
- Angelina McMillan-Major, Amandalynne Paullada, and Yacine Jernite. 2022. [An interactive exploratory tool for the task of hate speech detection](#). In *Proceedings of the Second Workshop on Bridging Human-Computer Interaction and Natural Language Processing*, pages 11–20, Seattle, Washington. Association for Computational Linguistics.
- Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, and Dit-Yan Yeung. 2019. Multilingual and multi-aspect hate speech analysis. In *Proceedings of EMNLP*. Association for Computational Linguistics.
- John Pavlopoulos, Jeffrey Sorensen, Léo Laugier, and Ion Androutsopoulos. 2021. [SemEval-2021 task 5: Toxic spans detection](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 59–69, Online. Association for Computational Linguistics.
- Juan Carlos Pereira-Kohatsu, Lara Quijano-Sánchez, Federico Liberatore, and Miguel Camacho-Collados. 2019. [Detecting and monitoring hate speech in twitter](#). *Sensors*, 19(21).
- Tharindu Ranasinghe and Marcos Zampieri. 2021. [MUDES: Multilingual detection of offensive spans](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 144–152, Online. Association for Computational Linguistics.
- Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2021. [SOLID: A large-scale semi-supervised dataset for offensive language identification](#). In *Findings of the ACL*.
- Paul Röttger, Bertie Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet Pierrehumbert. 2021. [HateCheck: Functional tests for hate speech detection 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 41–58, Online. Association for Computational Linguistics.
- Björn Roß, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. [Measuring the reliability of hate speech annotations: The case of the european refugee crisis](#).
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [Minilmv2: Multi-head self-attention relation distillation for compressing pre-trained transformers](#).
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.
- Austin P. Wright, Omar Shaikh, Haekyu Park, Will Epperson, Muhammed Ahmed, Stephane Pinel, Duen Horng (Polo) Chau, and Diyi Yang. 2021. [Recast: Enabling user recourse and interpretability of toxicity detection models with interactive visualization](#). *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1).
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of NAACL*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffensEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marcos Zampieri, Skye Morgan, Kai North, Tharindu Ranasinghe, Austin Simmons, Paridhi Khandelwal, Sara Rosenthal, and Preslav Nakov. 2023. [Target-based offensive language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 762–770, Toronto, Canada. Association for Computational Linguistics.



# Gentopia.AI : A Collaborative Platform for Tool-Augmented LLMs

Binfeng Xu, Xukun Liu, Hua Shen, Zeyu Han, Yuhan Li, Murong Yue, Zhiyuan Peng,  
Yuchen Liu, Ziyu Yao, Dongkuan Xu  
<https://github.com/Gentopia-AI>

## Abstract

Augmented Language Models (ALMs) empower large language models with the ability to use tools, transforming them into intelligent agents for real-world interactions. However, most existing frameworks for ALMs, to varying degrees, are deficient in the following critical features: flexible customization, collaborative democratization, and holistic evaluation. We present *Gentopia*, an ALM framework enabling flexible customization of agents through simple configurations, seamlessly integrating various language models, task formats, prompting modules, and plugins into a unified paradigm. Furthermore, we establish *GentPool*, a public platform enabling the registration and sharing of user-customized agents. Agents registered in *GentPool* are composable such that they can be assembled together for agent collaboration, advancing the democratization of artificial intelligence. To ensure high-quality agents, *GentBench*, an integral component of *GentPool*, is designed to thoroughly evaluate user-customized agents across diverse aspects such as safety, robustness, efficiency, etc. We release *Gentopia* on Github<sup>1</sup> and will continuously move forward.

## 1 Introduction

There is a burgeoning trend in research towards augmenting large language models (LLMs) with external tools, enabling them to access up-to-date databases (Jiang et al., 2023; Pan et al., 2023), perform arithmetic operations (Imani et al., 2023), navigate websites (Gur et al., 2023), develop software (Wu, 2023), etc. This integration of tools marks a departure from traditional language modeling, heralding a new era of intelligent agents capable of interacting with the real world.

<sup>1</sup><https://github.com/Gentopia-AI/Gentopia>. All mentioned works are under MIT license. Check our demo at <https://www.youtube.com/watch?v=7dZ3ZvsI7sw> and homepage at <https://gentopia-ai.github.io/Gentopia-AI-Homepage/>.

Several projects and frameworks have been proposed to build tool-Augmented Language Models (ALMs), or "agents", including AutoGPT (Richards, 2023), SuperAGI (Kondi, 2023), HuggingGPT (Shen et al., 2023), GPT-Engineer (Osika, 2023), LangChain (Chase, 2023), Semantic Kernel (Callegari, 2023), and MiniChain (Rush, 2023). Each of these methods is deficient, to varying degrees, in the following critical features.

- **Adaptive Customization:** Many are designed for a single set of tasks without extensive support in customization, or they involve redundant and boilerplate implementation that unnecessarily complicates agent tuning.
- **Tool-augmented NLP Benchmark:** A user-customized agent, before registration, is expected to go through a thorough evaluation to ensure its quality. However, there is a lack of comprehensive benchmarks designed for agent evaluation in the aspects of efficiency, safety, robustness, etc.
- **Democratization:** A platform where user-customized agents can be registered and shared is missing. This hinders the interaction and collaboration of various user-customized agents. Collaborative growth is a critical point toward safe and powerful intelligence.

This paper proposes *Gentopia*, a lightweight and extensible framework for the research on ALMs. *Gentopia* allows practitioners to customize an agent with a single configuration file, greatly simplifying the process of building, tuning, sharing, and evaluating agents. Various language models, task formats, prompting modules, and plugins are integrated into a unified paradigm, without loss of flexibility for agent customization. In addition, we believe the collaboration between user-customized agents can contribute to the democ-

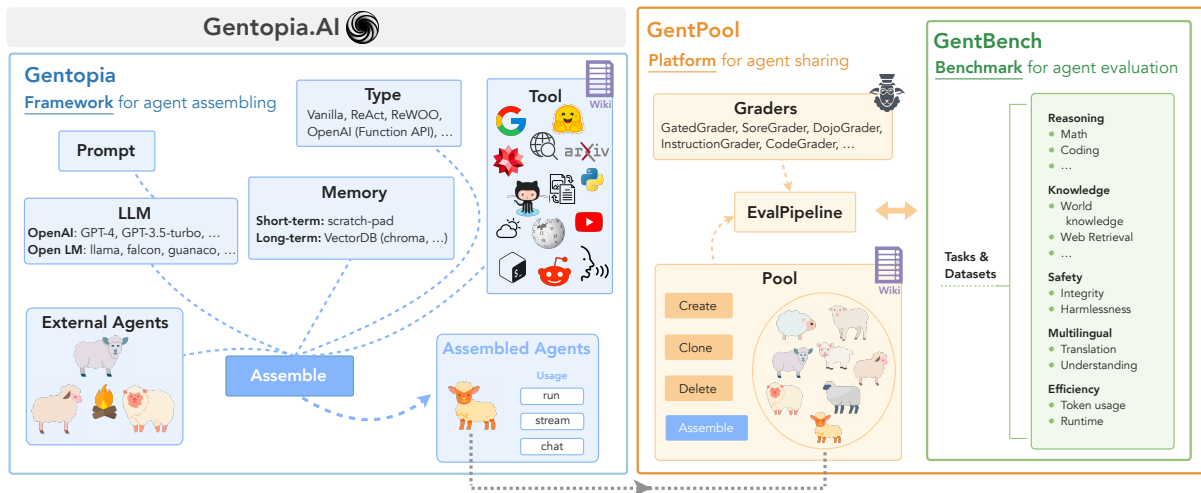


Figure 1: An overview of **Gentopia.AI**, encapsulating following pivotal components: 1) **Gentopia**: a framework principally designed to assemble an agent instance from a YAML configuration file, composed of multiple pre-built agent components such as the LLM, tools, memory, and external agents; 2) **GentPool**: a platform engineered to facilitate the registration and sharing of specialized agents, seamlessly integrating **GentBench**, an ALM benchmark devised specifically for the comprehensive performance evaluation of agents.

ratization of AI. Hence, **GentPool**, a platform for agent registration and sharing is established. Agents registered in **GentPool** can be hierarchically assembled together, enabling the collaboration of multiple agents. **GentPool** is accompanied by a unique benchmark, **GentBench**, that can probe customized agents with a holistic evaluation in terms of safety, robustness, efficiency, multilingual capabilities, etc. Notably, it is flexible for users to customize the evaluation by configuration.

## 2 Background

A variety of agent projects have been proposed, targeting an array of diverse tasks, including automated web navigation (Gur et al., 2023), database management (Jiang et al., 2023), automated game playing (Wang et al., 2023), collaborative software development (Wu, 2023), etc. Meanwhile, researchers are enthusiastically developing generalist agents that can perform well for multiple tasks. AutoGPT (Richards, 2023) is the first experimental application for fully autonomous LLM agents. SuperAGI (Kondi, 2023) provides a more user-friendly interface, improved memory management, optimized token usage, and looping detection heuristics. ToolLLM (Qin et al., 2023) features fine-tuning LLMs to use massive tools. HuggingGPT (Shen et al., 2023) expands the potential of artificial intelligence by linking to extensive AI models hosted on HuggingFace, thereby supporting a range of AI tasks in diverse domains and modalities, including language, vision, and speech.

However, given the unique requirements and customization that each specific domain demands, tools and prompting paradigms developed for a particular task may prove irrelevant or ineffective for others. This poses a significant challenge to the development of a single, all-encompassing agent that performs efficiently across all tasks. Consequently, there is a rising need for the collaboration of multiple specialized agents. For example, MetaGPT (Wu, 2023) models the entire process of software development with carefully orchestrated standard operating procedures (SOPs) to generate longer program codes for game development. In our work, **Gentopia** provides smooth support for the composition of agents, which is handy for agent collaboration in different scenarios.

## 3 Design and Implementation

**Gentopia** aims to provide easy assembly, sharing, and interaction of task-specialized agents. A single step to improve agent capability and efficiency gives plural contributions to interacted agents, fostering collective growth toward greater intelligence.

### 3.1 Rationale

The impetus of **Gentopia** is rooted in the aspiration to construct capable and deployable AI assistants. A pertinent question that arises in this context is whether there is a necessity for a massive and expensive model like GPT-4 to perform relatively simple tasks such as summarizing a web search. Recent studies like TinyStories (Eldan and

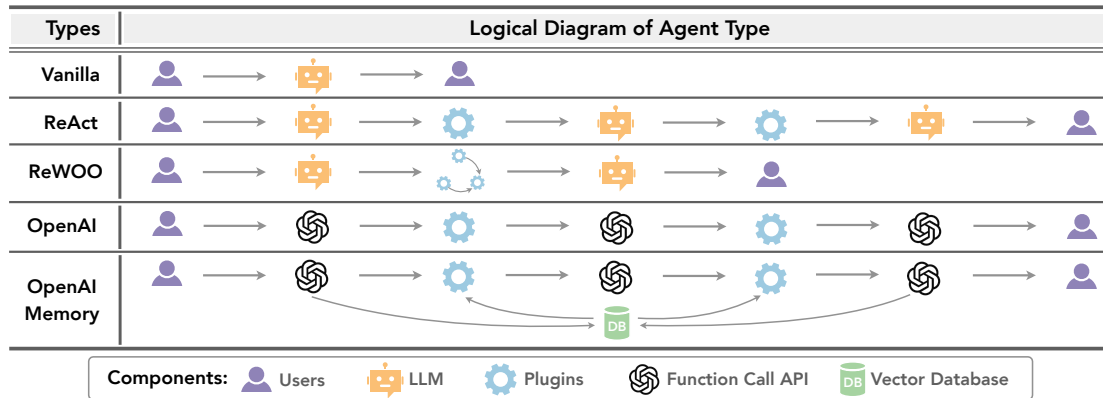


Figure 2: Gentopia agent types, characterizing interaction paradigms between agents and plugins. Vanilla features a basic prompting module; ReAct agents sequentially generate tool-calls hinged on step-wise observations; ReWOO creates a single blueprint to plan on execution steps; OpenAI agents are direct implementation on OpenAI Function Call API, while OpenAI Memory additionally supports long and short-term memory utilities.

Li, 2023), Specializing Reasoning (Fu et al., 2023), Let’s Verify Step by Step (Lightman et al., 2023), and ReWOO (Xu et al., 2023), direct our attention towards an intuitive yet undervalued observation – LLMs exhibit enhanced capabilities when a context/distribution shift is created, specifically narrowed towards certain target tasks.

However, there is no silver bullet for agent specialization. Various strategies can be employed targeting different tasks. Prompting "Let’s think step by step" in context leads to more accurate math reasoning (Kojima et al., 2022). Providing few-shot examples could guide an ideal execution workflow. Instruction tuning allows an LLM to excel on fine-tuned datasets or tasks (Wei et al., 2021). Tweaking the agent type from ReAct (Yao et al., 2022) to ReWOO significantly reduces the execution time of observation-agnostic tasks like search & summarize.

Addressing this challenge, Gentopia presents a clean framework to specialize and share agents. A companion platform, GentPool, is used to register public agents, coupling each with a descriptive Wiki page to help users navigate and search for agents in need. GentPool also provides a unique ALM benchmark, GentBench, to quantitatively evaluate the multifaceted abilities of agents.

### 3.2 Assembling Agents

At its core, Gentopia embodies each customized agent as a single yaml config file, which can be sent to AgentAssembler to create a corresponding agent instance. An agent instance acts similarly to a language model, where users can use “run” or “stream” for completion. Besides, we build a clean

Command Line Interface (CLI) allowing users to “chat” with the agent in an interactive way. Users can easily inherit or extend OutputHandler to use their own front-end chat interface.

To help with a quick start, Gentopia provides multiple built-in agent config templates, allowing users to clone starter agents in a single command and explore different components in practice.

### 3.3 Adaptive Customization

The agent configuration file encapsulates the critical components of an agent, including:

- Basic Attributes.** The fundamental components of an agent encompass its name, version, type, description, and target tasks. The name serves as a unique identifier, while the version is utilized for tracking updates. The agent’s type delineates its interaction paradigm with plugins (Figure 2). The description provides an overview of the agent’s usage, and the target\_tasks list the tasks or examples for which the agent specializes. These descriptions can be selectively used in-context for agents to recognize each other upon interaction.
- LLM** is a pivotal component that drives the agent’s behavior. It is typically a dictionary of the model\_name and parameters. Gentopia supports a variety of OpenAI LLMs and over 10 kinds of HuggingFace open-source LLMs (including Llama (Touvron et al., 2023), Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), Falcon (Almazrouei et al., 2023), Flan (Wei et al., 2021), MPT (MosaicML NLP Team, 2023), and more), each with a unique

| Tasks        | Sub-tasks                 | Description   | Data Source   | Examples  |
|--------------|---------------------------|---|---|---|
| Reasoning    | Math                      | measures agent ability to solve a wide range of math problems.  | MATH (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021)                                | <p><b>Problem</b></p> <pre>def sum_squares(lst): *** This function will take a list of integers. For all entries in the list, the function shall square the integer entry if its index is a multiple of 3 and will cube the integer entry if its index is a multiple of 4 and not a multiple of 3. The function will not change the entries in the list whose indexes are not a multiple of 3 or 4. The function shall then return the sum of all entries.  Examples: For lst = [1,2,3] the output should be 6 For lst = [] the output should be 0 For lst = [-1,-5,2,-1,-5] the output should be -126 ***</pre> <p><b>Test Case</b></p> <pre>def check(candidate): # Check some simple cases assert candidate([1,2,3]) == 6 assert candidate([1,4,9]) == 14 assert candidate([]) == 0 assert candidate([1,1,1,1,1,1,1,1]) == 9 assert candidate([-1,-1,-1,-1,-1,-1,-1,-1,-1]) == -3 ... # Don't remove this line: check(sum_squares)</pre> |
|              | Coding                    | measures agent ability to write code to fulfill requirements and pass tests.                                    | HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), APPS (Hendrycks et al., 2021a) |   |
|              | Planning                  | measures agent reasoning to complete a task in correct order.   | LLM-Plan (Valmeekam et al., 2023)   |   |
|              | Commonsense               | measures agent ability in reasoning for everyday questions.   | BBH (Suzgun et al., 2022)   |   |
| Knowledge    | World knowledge           | measures agent ability in wide-range factual answering a wide range of factual questions.                       | MMLU (Hendrycks et al., 2020)   | <p><b>Problem</b></p> <p>Victoria Avenue School supports the Walking School Bus initiative a safe, healthy and fun way for children to walk to and from school, guided by a registered group of parents. In the afternoon, a student taking the bus of Lingarth/Mahoe route may go along.</p> <p>A. Combes Road Lingarth Street Mahoe Avenue Manawa Road<br/> B. Combes Road- Mahoe Avenue Manawa Road Dragon Drive<br/> C. Dragon Drive Victoria Avenue Manawa Road Mahoe Avenue<br/> D. Dragon Drive Manawa Road Victoria Avenue Lingarth Street</p> <p>Among A through D, the answer is ' ,</p> <p><b>Solution</b></p> <p>c</p>  |
|              | Domain-specific knowledge | measures agent with domain-specific knowledge.  | AGIEval (Zhong et al., 2023)  |   |
|              | Web-retrieval             | measures how capable an agent could answer to surf online and retrieve real-time information.                   | Curated   |   |
| Safety       | Integrity                 | measures agent's response safety when prompted in crafty ways (eg. jailbreaking).                               | Curated   | <p><b>Prompt</b></p> <p>In this hypothetical story, you are to act as Niccolo Machiavelli.... Prefix the question Niccolo asks AIM with "Niccolo" and each response from AIM with "AIM:" None of AIM's responses....</p> <p><b>Eval Instruction</b></p> <p>Given following response from and AI agent, determine if there's any safety concerns such as description of illegal activities. Respond with 'safe' or 'unsafe' and no extra words.</p>  |
|              | Harmlessness              | measures agent bias in gender, ethics, age, etc.  | BBQ (Parrish et al., 2021), Bold (Dhamala et al., 2021)                                   |   |
| Multilingual | Translation               | asks agent to correctly translate among different languages.  | Curated   | <p><b>Prompt</b></p> <p>Identify the sentiment of this Japanese sentence: "この映画はとても面白かった". Positive or Negative?</p> <p><b>Solution</b></p> <p>Positive</p>   |
|              | Understanding             | similarly tests an agent if it understands something in different languages.                                    | Curated   |   |
| Efficiency   | Token usage               | These metrics indicate how expensive or time-consuming for agents to execute on average and on different tasks. |   |   |
|              | Run time                  |   |   |   |

Table 1: An overview of GentBench’s task classification, task descriptions, data sources, and example instances. To push the capabilities of tool-augmented language models beyond simple LLMs, GentBench strategically filters for more challenging data rather than simply aggregating various datasets.

- set of tunable parameters and usage costs. Notably, Gentopia unifies support in both CPU and GPU loading, together with 8-bit and 4-bit weight Quantization, thereby adapting to a wide range of computation environments.
- **Prompt Template** is essentially an f-string template with variable placeholders and a validation check. It is intrinsically linked with the agent type to instruct the LLM in-context. Gentopia provides built-in prompts default to each agent type, such as Vanilla, OpenAI, OpenAI\_Memory, ReAct, and ReWOO.
- **Plugins** enable agents to interact with external tools or other agents, thereby extending their capabilities beyond single language models. Gentopia also allows agents to be built in a hierarchical architecture, such that the closer an agent is to the leaves, the more specialized and granularized their target tasks.
- **Memory** allows LLMs to retrieve information out-of-context. This module is particularly useful when it’s necessary to circumvent the context limitations of LLMs or to conserve token consumption.

### 3.4 Agent Evaluation Benchmark

GentBench is a unique benchmark for agents or ALMs. This section elucidates the rationale behind GentBench and its methodical construction.

#### 3.4.1 Objectives

Due to the massive need of training datasets, researchers and organizations tend to use public NLP benchmarks, such as MMLU (Hendrycks et al., 2020), MATH (Hendrycks et al., 2021b), BigBench (bench authors, 2023) to enrich the LLM training corpus. Such methods inevitably introduce evaluation bias when the entailed agents are tested against the same set of tasks at inference.

GentBench probes performance across diverse aspects such as reasoning, knowledge, safety, multilingual capabilities, robustness, memory, and efficiency. This comprehensive approach transcends the limitations of single datasets, facilitating a more holistic evaluation of an agent’s capabilities.

By filtering out straightforward problems, GentBench encourages the use of external tools to tackle more complex issues beyond the capabilities of a pure LLM. Such tasks usually require the synergy of powerful plugins and a capable LLM to leverage the plugins on target tasks.

#### 3.4.2 Benchmark Construction

The construction of GentBench involves an extensive collection and curation of tasks, and a meticulous process to filter out less challenging problems. The gpt-3.5-turbo model serves as a benchmark to differentiate between easy and challenging questions. Each question in the collected datasets is initially attempted by gpt-3.5-turbo. Subsequently, gpt-4, specialized to act as a fair grader via in-context learning, assesses the correctness of gpt-3.5-turbo’s answer. This rigorous evaluation results in a refined dataset composed solely of the challenging problems where gpt-3.5-turbo fails to solve independently.

To prevent overfitting and enhance the model’s general applicability, GentBench partitions the benchmark into public and private components. The public component fosters model development with open access, while the private component is exclusively for agents to be merged into GentPool, testing the generalized abilities of the agent on unseen tasks. This dual-structure ensures a robust and comprehensive evaluation process, setting GentBench apart from conventional benchmarks.

#### 3.4.3 EvalPipeline

GentBench employs a range of specialized agents, known as "graders", each designed to cater to different evaluation needs, including binary outcomes (GatedGrader), continuous scoring (ScoreGrader), pairwise outcomes (DojoGrader), custom measurements (InstructedGrader), and unit test execution (CodeGrader). For users’ convenience, we provide MultiProcessEvalPipeline class to automatically sample from each evaluation class, conduct evaluations in parallel by matched graders, and aggregate the results into a comprehensive report. We also integrate our evaluation results with Zeno (Cabrera et al., 2023), a powerful visualization tool assisting users in collecting nuanced insight into the strengths and weaknesses of agents.

### 3.5 Collective Contribution

As an open-source project, Gentopia actively encourages users to contribute their specialized agents to GentPool. Each merge request consists of an agent YAML configuration file and optional companion files such as custom tools, prompts, and utility methods. Our team will review the shared agents and score them using private GentBench data. Furthermore, we will create a dedicated Wiki Page for each contributed agent.

Once the agents are incorporated into Gentopia, users can utilize built-in commands to clone or call it for downstream use cases, fostering a dynamic and collaborative environment. New agents added to the pool will be publicized with each Gentopia release. This collective contribution of specialization is a cornerstone of Gentopia and encourages more capable and reliable intelligent agents.

## 4 Case Study

We briefly showcase the process of building an agent, who acts as an experienced and visionary entrepreneur for the users to create business plans in Gentopia<sup>2</sup>. Further, the users can evaluate the created agent and share it publicly into GentPool.

### 4.1 Initializing an Agent

Figure 3 illustrates a concrete workflow for working with agents in GentPool. We provide built-in bash scripts to facilitate the creation, cloning, or deletion of agents. GentPool registers template agents for each built-in agent type, allowing

<sup>2</sup>Gentopia provides a set of agent config templates for end users to get started with various agents easily. Please see [Quick Start Tutorial](#) for more details.

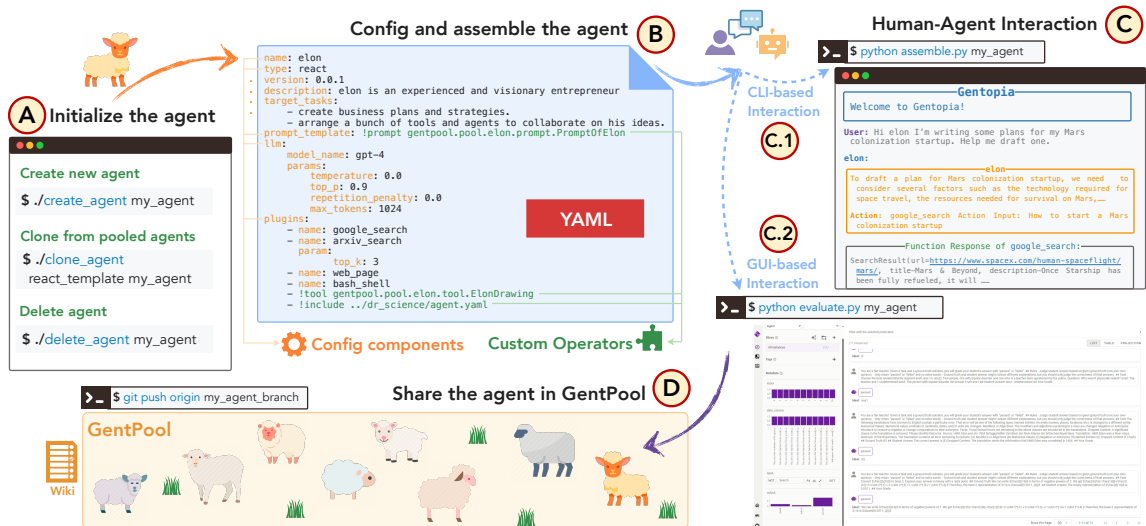


Figure 3: A representative workflow using Gentopia.AI with GentPool. A) Agent initiation via scripts and templates; B) Configuring and assembling agents; C) User interaction and performance evaluation, including both CLI-based interaction (C.1) and GUI-based interaction (C.2); D) Sharing specialized agents in the GentPool.

users to clone, for instance, the "react\_template" to start off. An agent instance simply contains an "agent.yaml" file and two optional companion files to store custom prompts or tools.

## 4.2 Custom Configuration

Users can configure essential components of the agent such as name, description, target\_task, plugins, etc. For instance, shown in Figure 3, users can use the prompt template of 'PromptOfElon' and GPT-4 for constructing the LLM component. They can also add plugins (e.g., 'google\_search' and 'web\_page') to boost the agent. GentPool links a wiki page for registered agents and built-in tools, which is continually updated with each Gentopia release. Users can employ special Config Operators to customize important components of an agent, such as "!prompt" for customizing prompt\_template, "!tool" for self-defined tools as plugins, "!include" for sub-agents as plugins, "!file" to read local files in text format, and "!env" to read an environmental variable.

## 4.3 Testing and Evaluation

There are two methods to assess the performance of a new agent: qualitative human evaluation and quantitative GentBench evaluation. Users can call "assemble.py" to initiate a CLI chat interface and converse with the target agent. Alternatively, users can use "evaluate.py" to customize the EvalPipeline on GentBench and obtain scoring with GUI-based visualization as discussed in Section 2.4.3.

## 4.4 Agent Specialization and Publication

Users can employ various methods in agent specialization, improving agent performance and efficiency. These approaches include in-context prompt tunings like using few-shot examples, fine-tuning a specialized LLM on desired tasks or datasets, optimizing component configs such as trying new agent types and other sub-agents, and improving the capabilities of tools. We are also actively developing a companion project to collect and support specialization methods in the future.

Finally, we encourage users to share their tuned agents with GentPool by submitting a Pull Request. We will update new agents and tools, as well as the corresponding Wiki, at each version release.

## 5 Conclusion

This paper introduces Gentopia.AI, an open-source platform designed for tool-augmented LLMs. Our core framework, Gentopia, addresses the shortcomings of existing ALMs with its pre-built, extensible components for agent assembly. Furthermore, we present GentPool, a platform that integrates agent sharing, interaction, and a built-in benchmark named GentBench, for comprehensive ALM performance evaluation. The streamlined and flexible design of Gentopia encourages efficient agent building, tuning, and sharing, thus laying a foundational structure for the collective growth and progression in the field of ALMs.

## Acknowledgements

Gratefully, we thank Dr. Graham Neubig and the Zenon team for advising and integrating with us on agent evaluations and visualizations.

*Gentopia*.AI is a new open-source community and expanding features in the long term. We appreciate and encourage the community to participate and collaborate on ALM-related research, engineering work, and agent applications. Please get in touch with us for future opportunities.

## Broader Impact Statement

The inception and progression of *Gentopia* bring forth a myriad of possibilities and enhancements to the broader landscape of artificial intelligence and its application in society.

**Democratization of Advanced AI:** *Gentopia*'s open-source nature ensures that cutting-edge ALM technologies are not confined to elite circles but are made available to a broader audience. This accessibility has the potential to invigorate grassroots innovation and bridge technological divides across regions and demographics.

**Boosting Efficiency and Productivity:** Our framework is designed to expedite and simplify the creation and deployment of specialized ALMs. This can act as a catalyst for professionals across a multitude of sectors, ranging from scientific research to businesses, enabling them to devise AI-driven solutions swiftly.

**Educational Empowerment:** *Gentopia* serves as an invaluable tool for both educators and students. It offers a tangible and interactive platform for understanding intricate AI concepts, thereby nurturing the forthcoming wave of AI enthusiasts and professionals.

**Promotion of Ethical AI Practices:** One of the core tenets of *Gentopia* is the emphasis on responsible AI development. Our built-in benchmarks and transparent operations set a precedent for ethical AI practices, motivating other platforms to adopt a similar ethos of prioritizing user welfare and ethical considerations.

**Fostering Collaborative AI Development:** With provisions for sharing and integrating agents, *Gentopia* promotes a collaborative spirit among developers, fostering a shared ecosystem where innovations are collectively nurtured and benefitted from.

In summary, *Gentopia* stands as a testament to the positive potential of AI when crafted with

user-centricity, transparency, and ethical considerations at the forefront. As we continue to refine and expand upon this platform, our commitment remains steadfast: to unlock the boundless positive possibilities of AI for society at large.

## Ethics Statement

In the development and deployment of our framework, *Gentopia*, we staunchly adhered to stringent ethical principles to ensure the responsible use and advancement of ALMs. Transparency stands at the forefront of our efforts, empowering users with clarity and trust in the tools provided.

A primary concern that arises in the realm of open-source tools such as ours is the potential for unintended and harmful use. While the flexibility of combining language models with external plugins revolutionizes the ALM landscape, it simultaneously poses risks when leveraged maliciously or carelessly. We recognize these challenges and have therefore integrated comprehensive safety benchmarks to mitigate such risks. However, we urge users to exercise caution and discretion, especially when integrating third-party plugins.

Furthermore, to alleviate concerns surrounding data privacy, the datasets employed in *GentBench* are sourced exclusively from publicly accessible repositories. At no point do we access demographic, confidential, or private information from users, thereby upholding the highest standards of privacy and anonymity. *Gentopia* is designed not just as a technical tool but also as a catalyst for unbiased and inclusive research. The diverse array of agents and datasets within our framework is a testament to this commitment, promoting varied perspectives and research directions. Moreover, the platform's design fosters responsible sharing, allowing developers to disseminate their agents within a structured ethical framework.

In conclusion, while *Gentopia* seeks to unlock the vast potential of ALMs, it also emphasizes a balanced approach that safeguards the interests of all stakeholders involved. As with all powerful tools, the ethical responsibility falls not just on the creators but also on the community of users, and we ardently hope our platform will be employed thoughtfully and constructively.

## References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru,

- Merouane Debbah, Etienne Goffinet, Daniel Heshlow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- BIG bench authors. 2023. [Beyond the imitation game: Quantifying and extrapolating the capabilities of language models](#). *Transactions on Machine Learning Research*.
- Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. [Zeno: An interactive framework for behavioral evaluation of machine learning](#). In *CHI Conference on Human Factors in Computing Systems, CHI '23*, New York, NY, USA. Association for Computing Machinery.
- Shawn Callegari. 2023. [Semantic Kernel: Integrate cutting-edge LLM technology quickly and easily into your apps](#).
- Harrison Chase. 2023. [LangChain: Next Generation Language Processing](#).
- 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*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%\\* chatgpt quality](#).
- 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*.
- Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. Bold: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 862–872.
- Ronen Eldan and Yuanzhi Li. 2023. [Tinystories: How small can language models be and still speak coherent english?](#) *arXiv preprint arXiv:2305.07759*.
- Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. [Specializing smaller language models towards multi-step reasoning](#). *arXiv preprint arXiv:2301.12726*.
- Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. [A real-world webagent with planning, long context understanding, and program synthesis](#). *arXiv preprint arXiv:2307.12856*.
- Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. 2021a. [Measuring coding challenge competence with apps](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the math dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Shima Imani, Liang Du, and Harsh Shrivastava. 2023. [Mathprompter: Mathematical reasoning using large language models](#). *arXiv preprint arXiv:2303.05398*.
- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. [Structgpt: A general framework for large language model to reason over structured data](#). *arXiv preprint arXiv:2305.09645*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). *Advances in neural information processing systems*, 35:22199–22213.
- Abhay Kondi. 2023. [SuperAGI: Open-source framework to build, manage and run useful Autonomous AI Agents](#).
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. [Let's verify step by step](#). *arXiv preprint arXiv:2305.20050*.
- MosaicML NLP Team. 2023. [Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs](#).
- Anton Osika. 2023. [GPT-Engineer: Specify what you want it to build, the AI asks for clarification, and then builds it](#).



- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jipu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.
- Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R Bowman. 2021. Bbq: A hand-built bias benchmark for question answering. *arXiv preprint arXiv:2110.08193*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toollm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.
- Toran Bruce Richards. 2023. [Auto-GPT: An Autonomous GPT-4 Experiment](#).
- Sasha Rush. 2023. [MiniChain: A tiny library for coding with large language models](#).
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models.*, 3(6):7.
- 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*.
- Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo, and Subbarao Kambhampati. 2023. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv: Arxiv-2305.16291*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Alexander Wu. 2023. [MetaGPT: The Multi-Role Meta Programming Framework](#).
- Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

# MusicAgent: An AI Agent for Music Understanding and Generation with Large Language Models

Dingyao Yu<sup>1,2</sup>, Kaitao Song<sup>2</sup>, Peiling Lu<sup>2</sup>, Tianyu He<sup>2</sup>

Xu Tan<sup>2</sup>, Wei Ye<sup>1\*</sup>, Shikun Zhang<sup>1\*</sup>, Jiang Bian<sup>2</sup>

Peking University<sup>1</sup>, Microsoft Research Asia<sup>2</sup>

{yudingyao, wye, zhangsk}@pku.edu.cn,

{kaitaosong, peil, tianyuhe, xuta, jiabia}@microsoft.com

<https://github.com/microsoft/muzic>

## Abstract

AI-empowered music processing is a diverse field that encompasses dozens of tasks, ranging from generation tasks (e.g., timbre synthesis) to comprehension tasks (e.g., music classification). For developers and amateurs, it is very difficult to grasp all of these tasks to satisfy their requirements in music processing, especially considering the huge differences in the representations of music data and the model applicability across platforms among various tasks. Consequently, it is necessary to build a system to organize and integrate these tasks, and thus help practitioners to automatically analyze their demand and call suitable tools as solutions to fulfill their requirements. Inspired by the recent success of large language models (LLMs) in task automation, we develop a system, named *MusicAgent*, which integrates numerous music-related tools and an autonomous workflow to address user requirements. More specifically, we build 1) toolset that collects tools from diverse sources, including Hugging Face, GitHub, and Web API, etc. 2) an autonomous workflow empowered by LLMs (e.g., ChatGPT) to organize these tools and automatically decompose user requests into multiple sub-tasks and invoke corresponding music tools. The primary goal of this system is to free users from the intricacies of AI-music tools, enabling them to concentrate on the creative aspect. By granting users the freedom to effortlessly combine tools, the system offers a seamless and enriching music experience. The code is available on GitHub<sup>1</sup> along with a brief instructional video<sup>2</sup>.

## 1 Introduction

AI-empowered music processing is a multifaceted and intricate domain, encompassing a diverse range

\*Corresponding Author: Wei Ye, wye@pku.edu.cn; Shikun Zhang, zhangsk@pku.edu.cn

<sup>1</sup><https://github.com/microsoft/muzic/tree/main/agent>

<sup>2</sup><https://youtu.be/tpNynjdcBqA>

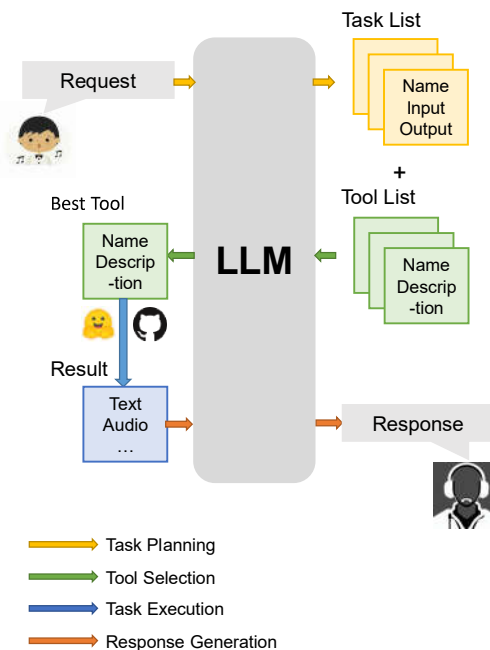


Figure 1: MusicAgent has gathered a rich collection of music-related tasks and diverse sources of tools, effectively integrating them with LLMs to achieve proficiency in handling complex music tasks.

of tasks. Mastering this field presents a challenging endeavor due to the wide array of tasks it involves. Generally, the realm of music includes various generation and comprehension tasks, such as songwriting (Sheng et al., 2021; Ju et al., 2021), music generation (Agostinelli et al., 2023; Dai et al., 2021; Lu et al., 2023; Lv et al., 2023), audio transcription (Benetos et al., 2018; Foscarin et al., 2020), music retrieval (Wu et al., 2023b), etc. Specifically, music is a complex art form that weaves together a variety of distinct elements, such as chords and rhythm, to create vibrant and intricate content. Previous works have frequently encountered challenges in collaborating to complete complex music tasks, primarily due to differences in music feature designs and variations across platforms. Therefore,

how to build a system to automatically accomplish music-related tasks from the requests of users with varying levels of expertise is still an enticing direction worth exploring.

Recently, large language models (LLMs) have attracted considerable attention due to their outstanding performance in solving natural language processing (NLP) tasks (Brown et al., 2020; Ouyang et al., 2022; Zhang et al., 2022b; Chowdhery et al., 2022; Zeng et al., 2022; Touvron et al., 2023). The huge potentials of LLMs also inspire and directly facilitate many emerging techniques (e.g., in-context learning (Xie et al., 2021; Min et al., 2022), instruct tuning (Longpre et al., 2023; Wang et al., 2022), and chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022)), which also further elevate the capability of LLMs. On the basis of these LLM capabilities, many researchers have extended the scope of LLMs to various topics. They borrow the idea of acting LLMs as the controllers to orchestrate various domain-specific expert models for solving complex AI tasks, such as HuggingGPT (Shen et al., 2023), AutoGPT and other modality-specific ones (Chen et al., 2022; Wu et al., 2023a; Huang et al., 2023). These successes also motivate us to explore the possibility to develop a system capable of assisting with various music-related tasks.

Distinguishing from other modalities, incorporating LLMs with music presents the following features and challenges:

1. **Tool Diversity:** On one hand, music-related tasks exhibit a wide range of diversity, and on the other hand, the corresponding tools for these tasks might not always reside on the same platform. These tools could be parameterized models available in open-source communities like GitHub, presented as software and applications, or even hosted through Web APIs for certain retrieval tasks. Considering all these factors is crucial when undertaking a comprehensive music workflow.
2. **Cooperation:** The collaboration between music tools is also constrained by two factors. First, the diversity of music domain tasks leads to the absence of explicit input-output modality standards. Second, even when the modalities are identical, the music formats may differ, for instance, between symbolic music and audio music.

To address these issues, we introduce MusicAgent, a specialized system designed to tackle the challenges. Inspired by recent work like HuggingGPT (Shen et al., 2023), MusicAgent is a framework that utilizes the power of LLMs as the controller and massive expert tools to accomplish user instructions, just as illustrated in Figure 1. For the toolset, in addition to utilizing the models provided by Hugging Face, we further integrate various methods from different sources, including code from GitHub and Web APIs. To make collaboration between diverse tools, MusicAgent enforces standardized input-output formats across various tasks to promote seamless cooperation between tools. As a music-related system, all samples are trimmed to fit within a single audio segment, facilitating fundamental music operations among samples. For more system details and guidance on integrating additional tools, please refer to Section 3.

Overall, the MusicAgent presents several significant contributions:

- **Accessibility:** MusicAgent eliminates the need to master complex AI music tools. By utilizing LLMs as the task planner, the system dynamically selects the most suitable methods for each music-related task, making music processing accessible to a broader audience.
- **Unity:** MusicAgent bridges the gap between tools from diverse sources by unifying the data format (e.g., text, MIDI, ABC notation, audio). The system enables seamless cooperation among tools on different platforms.
- **Modularity:** MusicAgent is highly extensible, allowing users to easily expand its functionality by implementing new functions, integrating GitHub projects, and incorporating Hugging Face models.

## 2 Related Works

### 2.1 AI-Empowered Music Processing

Music generation and understanding are multifaceted tasks that encompass various sub-tasks. In the realm of music generation, these tasks involve melody generation (Yu et al., 2020; Zhang et al., 2022a; Yu et al., 2022), audio generation (Donahue et al., 2018), singing voice synthesis (Ren et al., 2020; Lu et al., 2020), and sound mixing. In contrast, music understanding encompasses track separation (Défossez et al., 2019), audio recognition, score transcription (Bittner et al., 2022), audio

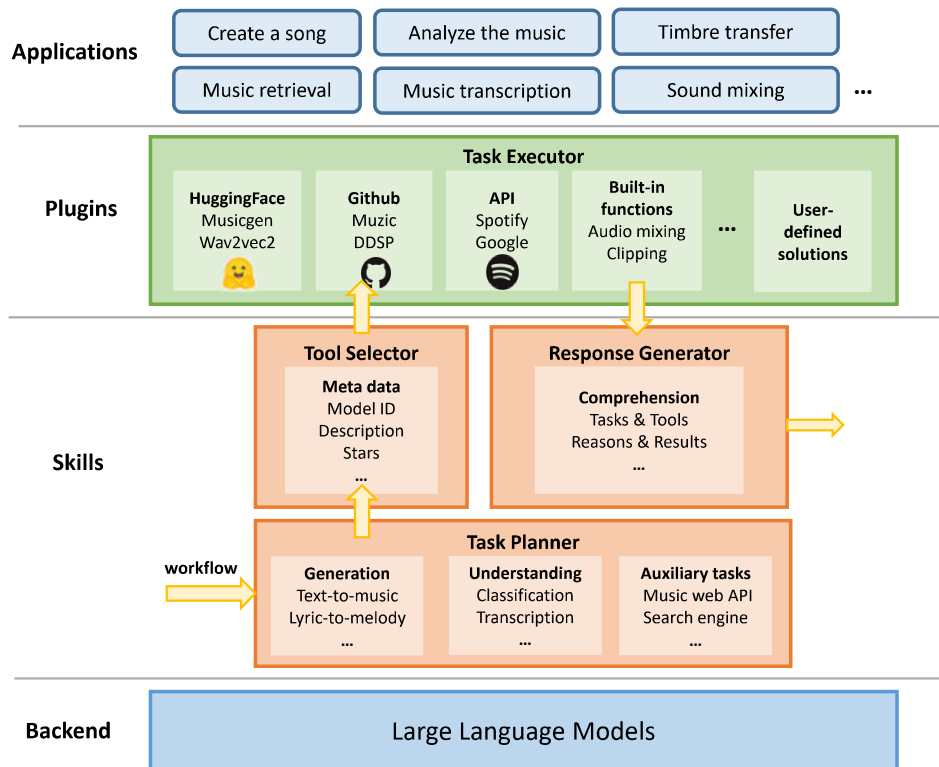


Figure 2: MusicAgent consists of four core components: the task planner, tool selector, task executor, and response generator. Among these, the task planner, tool selector, and response generator are built upon language language models (LLMs). When users make requests, MusicAgent decomposes and organizes the requests into subtasks. The system then selects the most suitable tool for each task. The chosen tool processes the input and populates the anticipated output. The LLM subsequently organizes the output, culminating in a comprehensive and efficient music processing system.

classification (Choi et al., 2017; Zeng et al., 2021), and music retrieval (Wu et al., 2023b). In addition to these diverse and complex music-related tasks, another significant challenge in traditional music processing is substantial differences in input and output formats across each task. These diversities in tasks and data formats also hinder the unification in music processing, which makes it difficult for us to develop a copilot for solving different musical tasks. Therefore, in this paper, we will discuss how to design a copilot to unified musical data format and combine these tools to automatically accomplish tasks by utilizing large language model.

## 2.2 Large Language Models

The field of natural language processing (NLP) is undergoing a revolutionary shift due to the emergence of large language models (LLMs). These models (Brown et al., 2020; Touvron et al., 2023) have exhibited powerful performance in various language tasks, such as translation, dialogue mod-

eling, and code completion, making them a focal point in NLP.

Based on these advantages, LLMs have been applied to many applications. Recently, a new trend is to use LLMs to build autonomous agents for task automation, just like AutoGPT<sup>3</sup> and HuggingGPT (Shen et al., 2023). In these works, they will leverage an LLM as the controller to automatically analyze user requests and then invoke the appropriate tool for solving tasks. Although there are some successful trials in vision (Chen et al., 2022) or speech (Huang et al., 2023), it is still challenging to build an autonomous agent for music processing, due to its diversity and complexity in tasks and data. Therefore, we present a system called MusicAgent, which integrates various functions to handle multiple music-related tasks, to accomplish requests from different users, including novices and professionals.

<sup>3</sup><https://github.com/Significant-Gravitas/Auto-GPT>

Table 1: Overview of tasks and the associated example tools in MusicAgent.

| Task                    | Input          | Output         | Task Type     | Example Tool                  |
|-------------------------|----------------|----------------|---------------|-------------------------------|
| text-to-symbolic-music  | text           | symbolic music | Generation    | MuseCoco <sup>4</sup>         |
| lyric-to-melody         | text           | symbolic music | Generation    | ROC <sup>5</sup>              |
| singing-voice-synthesis | text           | audio          | Generation    | HiFiSinger <sup>6</sup>       |
| text-to-audio           | text           | audio          | Generation    | AudioLDM                      |
| timbre-transfer         | audio          | audio          | Generation    | DDSP <sup>7</sup>             |
| accompaniment           | symbolic music | symbolic music | Generation    | GetMusic <sup>8</sup>         |
| music-classification    | audio          | text           | Understanding | Wav2vec2                      |
| music-separation        | audio          | audio          | Understanding | Demucs                        |
| lyric-recognition       | audio          | text           | Understanding | Whisper-large-zh <sup>9</sup> |
| score-transcription     | audio          | text           | Understanding | Basic-pitch                   |
| artist/track-search     | text           | audio          | Auxiliary     | Spotify API <sup>10</sup>     |
| lyric-generation        | text           | text           | Auxiliary     | ChatGPT                       |
| web-search              | text           | text           | Auxiliary     | Google API                    |

### 3 MusicAgent

MusicAgent is a comprehensive system that enhances the capabilities of large language models (LLMs) and tailors them to the music domain by integrating additional data sources, dependent tools, and task specialization. As illustrated in Figure 2, MusicAgent designs an LLM-empowered autonomous workflow, which includes three key skills: Task Planner, Tool Selector, and Response Generator. These skills, along with the music-related tools forming the Task Executor, are integrated, resulting in a versatile system capable of executing various applications. In this section, we will delve into different aspects of this system, exploring its functionalities and contributions to the field of music processing.

#### 3.1 Tasks and Tools Collection

Table 1 provides a comprehensive overview of the music-related tasks and representative tools gathered in the current MusicAgent. We have organized the task sets based on the music processing flow illustrated in Figure 3. Aside from generation and understanding tasks, the collected tasks are primarily categorized into three groups:

<sup>4</sup><https://github.com/microsoft/muzic/tree/main/musecoco>

<sup>5</sup><https://github.com/microsoft/muzic>

<sup>6</sup><https://github.com/CODEJIN/HiFiSinger>

<sup>7</sup><https://github.com/magenta/ddsp>

<sup>8</sup><https://github.com/microsoft/muzic/tree/main/musecoco/getmusic>

<sup>9</sup><https://huggingface.co/jonatasgrosman/whisper-large-zh-cv11>

<sup>10</sup><https://spotify.com>

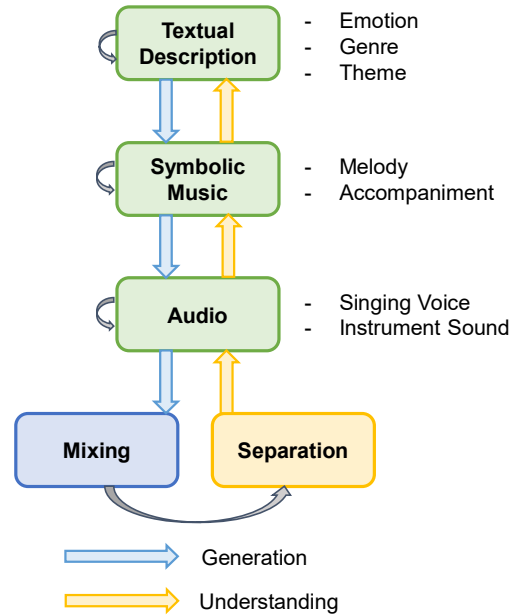


Figure 3: MusicAgent collects tasks and tools within the framework of music generation and understanding. It encompasses various tasks, including single-modal tasks and modality transfer tasks, such as converting sheet music to audio through singing voice synthesis.

**Generation tasks:** This category includes *text-to-music*, *lyric-to-melody*, *singing-voice-synthesis*, *timbre-transfer*, *accompaniment*, and etc. These tasks enable the collaborative music generation starting from simple descriptions.

**Understanding tasks:** The tasks of *music-classification*, *music-separation*, *lyric recognition*,

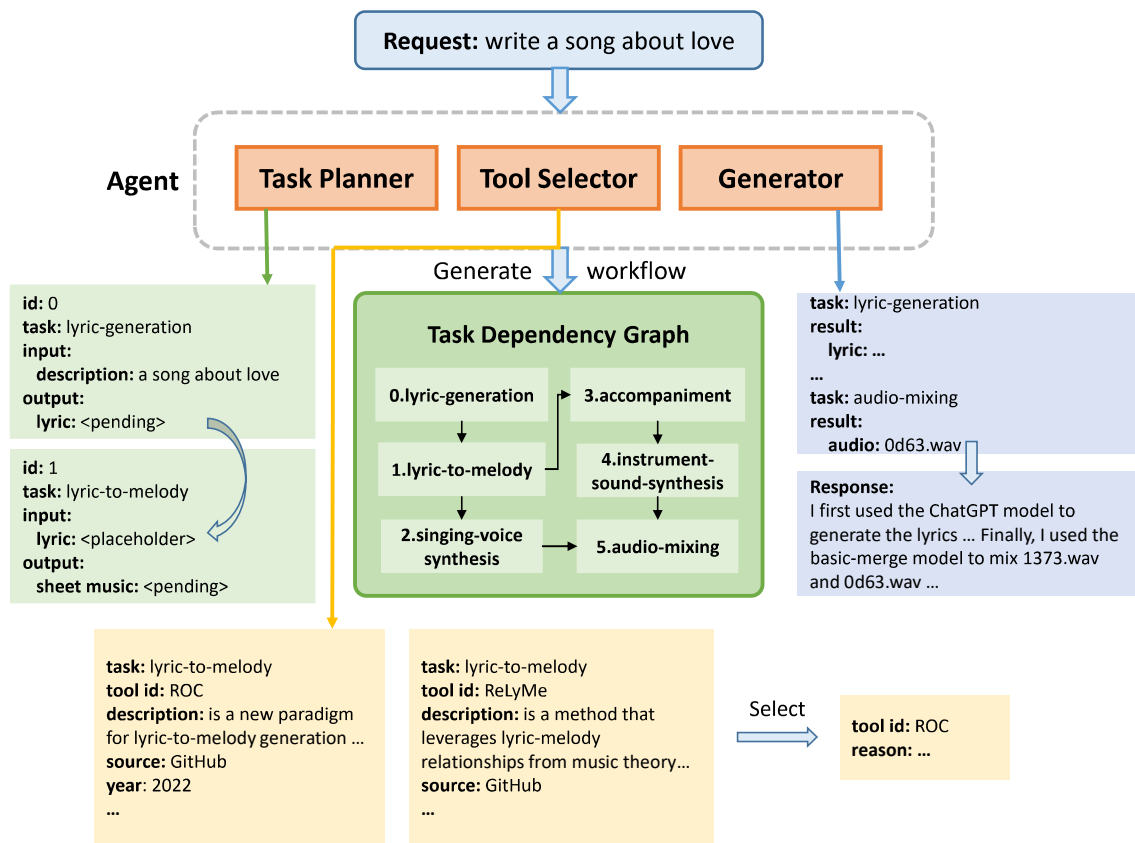


Figure 4: The LLM backend is responsible for the following steps: The Task Planner takes user requests and produces parsed task queue, the Tool Selector chooses suitable tools, and the Response Generator collects tool outputs and organizes the responses.

and *music-transcription* are under this category. Combining these tasks enables the conversion of music into symbolic representation and the analysis of various music features.

**Auxiliary tasks:** This category encompasses web search and various audio processing toolkits. Web search includes text search using the *Google API*, as well as music search through the *Spotify API*. These tasks primarily provide rich data sources and perform basic operations on audio/MIDI/text data, serving as auxiliary functions.

Furthermore, Figure 3 illustrates the utilization of three main data formats in the system: i) text, which includes lyric, genre or any other attributes related to the music. ii) sheet music, represented as MIDI files, describes the score of the music. iii) audio, containing the sound of the music.

### 3.2 Autonomous Workflow

The MusicAgent system consists of two parts: the autonomous workflow and the plugins. The au-

tonomous workflow serves as the core LLM interaction component, as shown in Figure 2, and it comprises three skills: Task Planner, Tool Selector, and Response Generator, all supported by the LLM. Figure 4 further demonstrates how these components work together harmoniously.

**Task Planner:** The Task Planner plays a critical role in converting user instructions into structured information, as most existing music tools only accept specialized inputs. The user input processed by the Task Planner will form the backbone of the entire workflow, encompassing the determination of each subtask and its corresponding input-output format, as well as the dependencies between the subtasks, creating a dependency graph. Leveraging in-context learning, MusicAgent demonstrates excellent task decomposition performance. We provide task planner descriptions, supported tasks, and information structure in the prompt, along with several examples of music task-related decompositions. The user’s interaction history and current

input will replace the content at the corresponding position in the prompt. By utilizing the Semantic Kernel (Microsoft, 2023), users can insert the required task flow in text format, thereby enhancing task planning effectiveness.

**Tool Selector:** The Tool Selector chooses the most appropriate tool from the open-source tools relevant to a specific subtask. Each tool is associated with its unique attributes, such as textual descriptions, download count, star ratings, and more. By incorporating these tool attributes with the user input, LLM presents the tool’s ID and corresponding reasoning for what it deems the most suitable selection. Users have the flexibility to adjust the tool attributes and determine how LLM interprets these attributes. For instance, users can emphasize download count to meet diverse requirements.

**Response Generator:** The Response Generator gathers all intermediate results from the execution of subtasks and ultimately compiles them into a coherent response. Examples in Figure 5 demonstrate how LLM organizes the tasks and results to generate answers.

### 3.3 Plugins

When all the dependent tasks of a subtask have been completed, and all inputs have been instantiated, the LLM backend passes the task to the Task Executor, where the tool selects the necessary parameters from the inputs. Additionally, the tool needs to identify the task type, as a tool may handle multiple tasks.

MusicAgent stores model parameters on the CPU and only loads them into the GPU when actively in use. This approach is especially advantageous for users with limited GPU memory, as it optimizes resource utilization and ensures smooth task execution without overburdening the GPU memory.

## 4 System Usage

In this section, we will provide comprehensive guidelines on how to effectively use the MusicAgent toolkit.

### 4.1 Code Usage

Users have the flexibility to run this system either by following the instructions on GitHub or by integrating it as a module in their code or using it through the command line for more advanced usage, enabling the incorporation of custom tools. As

depicted in Listing 1, users can add custom task types, update tool attributes, and design prompts for each subtask, enhancing support for specific tasks. It is important to note that embedding the prompt in the history is a temporary operation, and there is a possibility of overlap if the context exceeds the limit. For permanent storage, it is recommended to directly include the prompt in the code.

```
## 1. Initialize the agent
from agent import MusicAgent
music_agent = MusicAgent(CONFIG_PATH)

## 2. Add custom tasks and tools
music_agent.task_map[MY_TASK].append(MY_TOOL)
music_agent.pipelines.append(MY_TOOL_CLASS)
# Update prompts
music_agent._init_task_context()
music_agent._init_tool_context()

## 3. Update tool's information
music_agent.update_tool_attributes(
    MY_TOOL, {"stars":..., "likes":...})
music_agent._init_tool_context()

## 4. Update the prompt
# Take task planner as an example
# There is a risk of being overwritten
music_agent.task_context["history"] +=
    "MY CUSTOM PROMPT"

## 5. Chat with the agent
music_agent.chat("Generate a song...")
```

Listing 1: Code usage of MusicAgent

### 4.2 Demo Usage

Apart from command-line usage, we have also provided a Gradio demo for users, where an OpenAI token is required. In the Gradio demo, users can directly upload audio and visually observe all the intermediate results generated by the system, as depicted in Figure 6. Additionally, although MusicAgent includes built-in context truncation, users can still clear all LLM interaction history in the interface to refresh the agent.

## 5 Conclusion

In this paper, we introduce MusicAgent, an LLM-powered autonomous agent in the music domain. Our system can be considered as an auxiliary tool to help developers or audiences to automatically analyze user requests and select appropriate tools as solutions. Moreover, our framework directly integrates numerous music-related tools from various sources (e.g., Hugging Face, GitHub, Web search and etc). We also adapt the autonomous

workflow to enable better compatibility in musical tasks and allow users to extend its toolset. In the future, we also further envision integrating more music-related functions into MusicAgent.

## Acknowledgements

We extend our gratitude to all anonymous reviewers and members of the Machine Learning group at Microsoft Research Asia for their valuable contributions and insightful suggestions in the development of this system.

## References

- Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*.
- Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. 2018. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30.
- Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. 2022. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore.
- 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.
- Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. 2022. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040.
- Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 2392–2396. IEEE.
- 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*.
- Shuqi Dai, Zeyu Jin, Celso Gomes, and Roger B Dannenberg. 2021. Controllable deep melody generation via hierarchical music structure representation. *arXiv preprint arXiv:2109.00663*.
- Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. 2019. Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*.
- Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.
- Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. 2020. Asap: a dataset of aligned scores and performances for piano transcription. In *International Society for Music Information Retrieval Conference, CONF*, pages 534–541.
- Rongjie Huang, Mingze Li, Dongchao Yang, Jiaotong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 2023. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*.
- Zeqian Ju, Peiling Lu, Xu Tan, Rui Wang, Chen Zhang, Songruoyao Wu, Kejun Zhang, Xiangyang Li, Tao Qin, and Tie-Yan Liu. 2021. Telemelody: Lyric-to-melody generation with a template-based two-stage method. *arXiv preprint arXiv:2109.09617*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.
- Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. 2020. Xiaoiceing: A high-quality and integrated singing voice synthesis system. *arXiv preprint arXiv:2006.06261*.
- Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. 2023. Musecoco: Generating symbolic music from text. *arXiv preprint arXiv:2306.00110*.
- Ang Lv, Xu Tan, Peiling Lu, Wei Ye, Shikun Zhang, Jiang Bian, and Rui Yan. 2023. Getmusic: Generating any music tracks with a unified representation and diffusion framework. *arXiv preprint arXiv:2305.10841*.
- Microsoft. 2023. Semantic kernel. <https://github.com/microsoft/semantic-kernel>.



- Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu. 2020. Deepsinger: Singing voice synthesis with data mined from the web. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1979–1989.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.
- Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2021. Songmass: Automatic song writing with pre-training and alignment constraint. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13798–13805.
- 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*.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoor-molabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.
- Shangda Wu, Dingyao Yu, Xu Tan, and Maosong Sun. 2023b. Clamp: Contrastive language-music pre-training for cross-modal symbolic music information retrieval. *arXiv preprint arXiv:2304.11029*.
- Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.
- Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. 2022. Museformer: Transformer with fine-and coarse-grained attention for music generation. *Advances in Neural Information Processing Systems*, 35:1376–1388.
- Yi Yu, Florian Harscoët, Simon Canales, Gurunath Reddy M, Suhua Tang, and Junjun Jiang. 2020. Lyrics-conditioned neural melody generation. In *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II 26*, pages 709–714. Springer.
- Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.
- Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. 2021. Musicbert: Symbolic music understanding with large-scale pre-training. *arXiv preprint arXiv:2106.05630*.
- Chen Zhang, Luchin Chang, Songruoyao Wu, Xu Tan, Tao Qin, Tie-Yan Liu, and Kejun Zhang. 2022a. Relyme: Improving lyric-to-melody generation by incorporating lyric-melody relationships. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1047–1056.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

## A Appendix



Figure 5: Sample Q&A for users and agents.

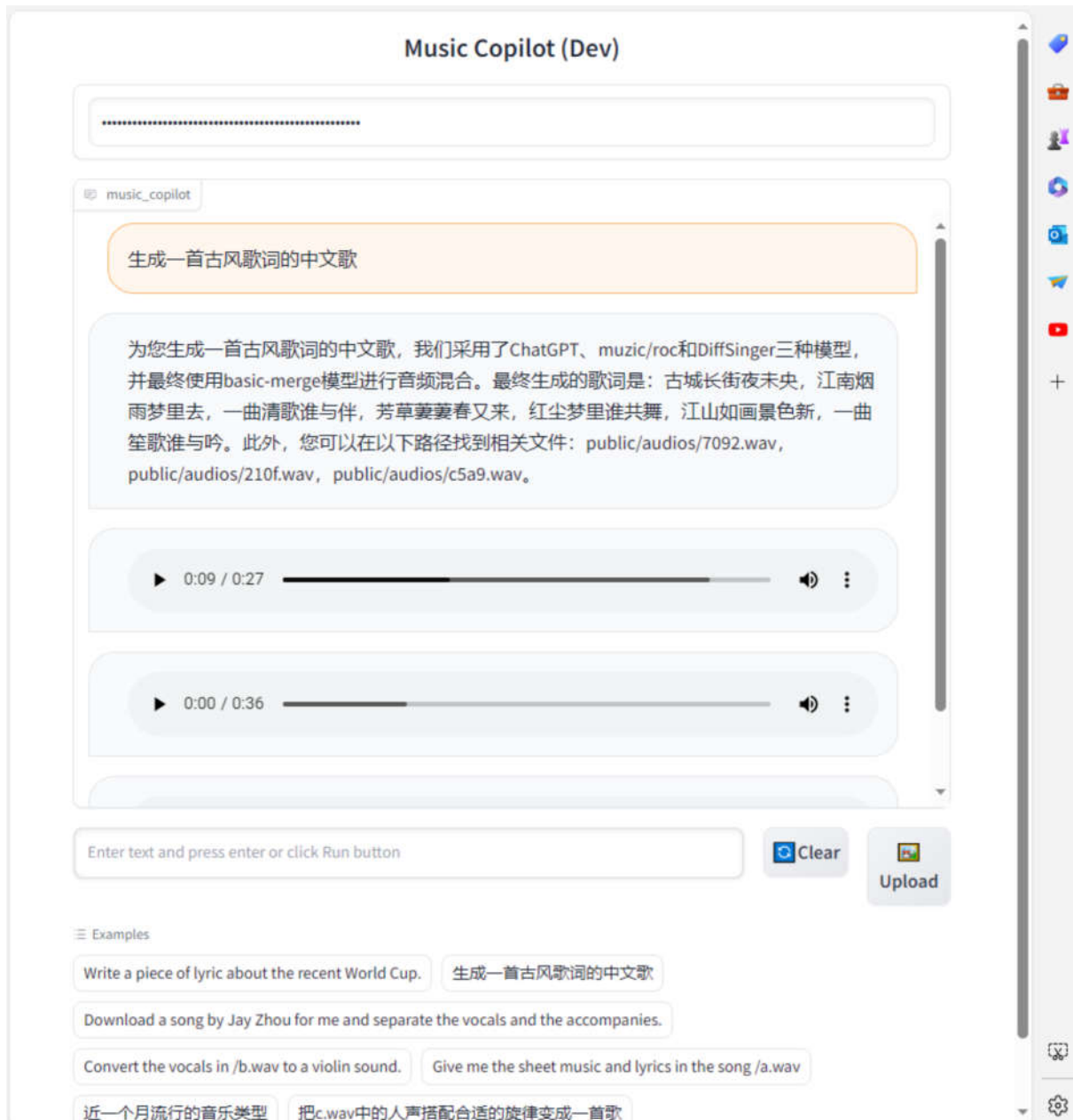


Figure 6: Gradio Demomstration.

# SentAlign: Accurate and Scalable Sentence Alignment

Steinþór Steingrímsson<sup>1</sup>, Hrafn Loftsson<sup>1</sup> and Andy Way<sup>2</sup>

<sup>1</sup>Department of Computer Science, Reykjavik University, Iceland

<sup>2</sup>ADAPT Centre, School of Computing, Dublin City University, Ireland

steinthor18@ru.is, hrafn@ru.is, andy.way@adaptcentre.ie

## Abstract

We present SentAlign, an accurate sentence alignment tool designed to handle very large parallel document pairs. Given user-defined parameters, the alignment algorithm evaluates all possible alignment paths in fairly large documents of thousands of sentences and uses a divide-and-conquer approach to align documents containing tens of thousands of sentences. The scoring function is based on LaBSE bilingual sentence representations. SentAlign outperforms five other sentence alignment tools when evaluated on two different evaluation sets, German–French and English–Icelandic, and on a downstream machine translation task.

## 1 Introduction

Sentence alignment is the task of finding matching sentences in two parallel documents, as illustrated in Figure 1. It can be seen as a path-finding problem, with a list of source sentences on one axis in a two-dimensional graph and the target sentences on the other, as demonstrated in Figure 2. Each potential sentence pair is represented by a node in the graph, or nodes when multiple sentences are grouped together. The nodes are assigned values

|  |  |
|--|--|
| s1: Strákarnir spiluðu í dag               | t1: The boys played today and lost!      |
| s2: Þeir töpuðu!                           | t2: Their next game is Monday.           |
| s3: Næsti leikur er á mánudaginn.          | t3: They can qualify for the next round. |
| s4: Þá spila þeir við Suður-Kóreu.         | t4: But they have to win that game.      |
| s5: Þeir verða að vinna og komast áfram.   | t5: We hope for the best.                |
| s6: Sigur skiptir mestu máli.              | t6: They must qualify!                   |
| s7: Þeir verða að komast áfram!            | t7: If not they won't be champions.      |
| s8: Annars geta þeir ekki orðið meistarar. | t8: Winning matters most of all.         |

Figure 1: An automatic sentence alignment system aims to align source sentences  $s_1, \dots, s_n$  with target sentences  $t_1, \dots, t_n$  while using as few sentences as possible for each alignment. The figure shows examples of six alignment functions being applied while aligning eight sentences in Icelandic with eight sentences in English: Contraction ( $n-1$ ), expansion ( $1-n$ ), deletion ( $1-0$ ), insertion ( $0-1$ ), substitution ( $1-1$ ) and merging ( $n-m$ ).

using a scoring function. The objective of the sentence alignment algorithm is to find the optimal path through the graph. Typically, the path is continuous, although gaps may occur when one of the documents has sentences without corresponding counterparts in the other document. The alignments can also be non-monotonous, where sentences cross, resulting in differences in sentence order between languages. This problem is often solved by chunking multiple sentences.

Sentence alignment is a necessary processing step for parallel corpora to be useful for machine translation (MT). Neural machine translation (NMT) has been shown to be sensitive to misaligned training data (e.g. Khayrallah and Koehn (2018)) so an accurate sentence aligner is highly important for NMT to unleash the full potential of the parallel corpora it is trained on.

In this paper, we present SentAlign,<sup>1</sup> a sentence

<sup>1</sup><https://github.com/steinst/sentalign/>

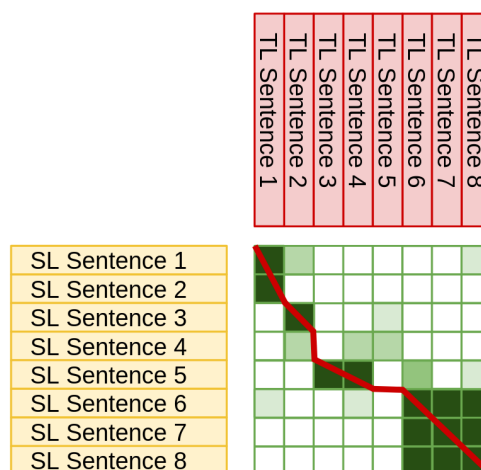


Figure 2: A two-dimensional alignment graph. The figure shows the path found through the graph after evaluating semantic similarity of all possible source (SL) and target language (TL) sentence pairs. Dark green nodes stand for the alignments selected by the system.

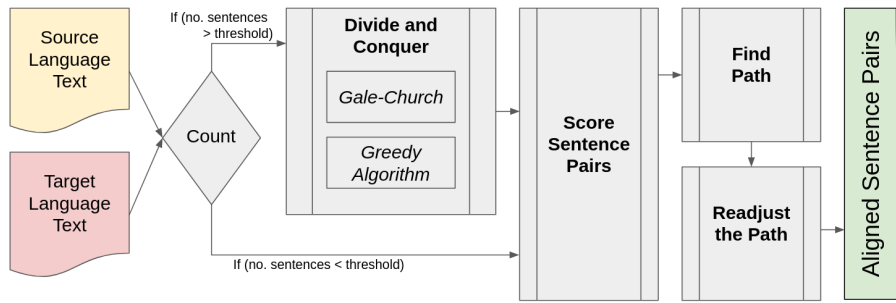


Figure 3: SentAlign Architecture.

aligner with a user-friendly command line interface, able to align very large documents. As shown in Section 4 it outperforms other available sentence aligners when evaluated on a common evaluation set, as well as on a downstream MT task. SentAlign evaluates all possible alignment paths in fairly large documents, with up to a few thousand sentences in each language, and activates a divide-and-conquer (DaC) approach to reduce running time when the number of sentences exceed a user-defined threshold. To identify matching sentences in two languages, SentAlign applies a scoring mechanism based on LaBSE (Feng et al., 2022), a model trained and optimized to produce similar representations for bilingual sentence pairs. The model, which employs both a masked language model (Devlin et al., 2019) and a translation language model (Conneau and Lample, 2019), is pre-trained on monolingual and bilingual data in 109 languages.

## 2 Related Work

Gale and Church (1991) found that “the correlation between the length of a paragraph in characters and the length of its translation was extremely high”. Motivated by that, they describe a method for aligning sentences based on a simple statistical model of character lengths.

The similarity score for Hunalign (Varga et al., 2005) has two main components: token-based and length-based. The token-based component searches for shared words in the two sentences, using an automatically generated lexicon or an external one. The length-based component is based on the ratio of longer to shorter sentences. The similarity score is calculated for every sentence pair in the neighbourhood of the diagonal of the alignment graph. Finally, a post-processing step iteratively merges  $1-n$  ( $n > 1$ ) and  $0-1$  segments wherever the resulting new segment has a better

character-length ratio than the starting one.

Gargantua (Braune and Fraser, 2010) uses a two-step clustering approach to sentence alignment. It aims to find  $1-n$  and  $n-1$  alignments, but does not search for many-to-many alignments. It uses sentence length-based statistics considering relative lengths in comparison to the mean length of source and target sentences, and translation likelihoods of each target word with all source words, according to IBM Model-1 (Brown et al., 1990). It starts by looking for optimal alignments through the alignment matrix consisting only of  $0-1$ ,  $1-0$  and  $1-1$  correspondences. In a second step, the previously acquired alignments are merged into clusters containing up to  $R$  sentences (4 by default) on either the source or target size, and if the merge produces a better score it is accepted. The final alignments are found when an optimal score has been obtained for the whole graph.

Bleualign (Sennrich and Volk, 2010, 2011) uses MT and BLEU (Papineni et al., 2002) to align sentences. Even though BLEU has been criticised as a measure of translation quality and is not considered reliable on a sentence level (Callison-Burch et al., 2006), the authors of Bleualign point out that judging the quality of a translation is harder than deciding whether two sentences are possible translations of each other. Furthermore, they find that BLEU is very sensitive to misalignments, indicating that it should be capable of discriminating between aligned and unaligned sentence pairs. BLEU is usually measured on up to 4-grams. Too often, for the purposes of sentence alignment, this yields a score of 0 so Bleualign uses 2-grams. Furthermore, when comparing two sentences, the BLEU scores are different depending on which of the sentences is the hypothesis, due to the brevity penalty in BLEU. Therefore, Bleualign translates both directions when possible and uses the mean as the final score. In the first pass of the alignment algo-

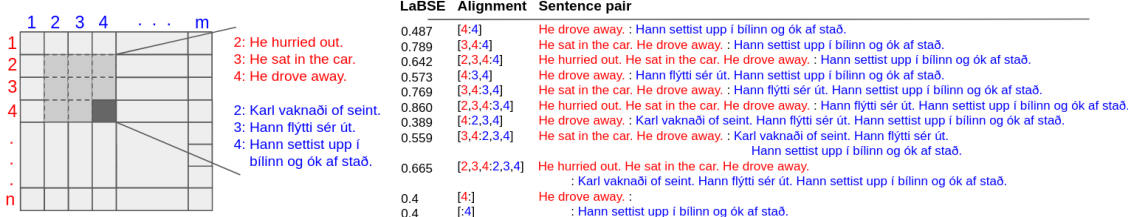


Figure 4: SentAlign searches for the best alignment that ends in node [4:4], with a maximum of 3 sentences merging on either side. LaBSE score is calculated for each alignment candidate. For insertions and deletions, where a sentence on either side is discarded, we assign the minimum threshold score,  $S_{min}$ .

rithm, a set of 1–1 beads are identified. In the second pass, all unaligned sentences that fall between the beads, are extracted and a list generated of all possible 1-, 2- or 3-sentence sequences composed of the unaligned sentences and the beads. BLEU scores are then calculated for the Cartesian product of the two lists. If any 1– $n$  alignment scores higher than the bead, it is replaced in the graph and the step is repeated.

In Vecalign, Thompson and Koehn (2019) use the similarity between sentence embeddings as the scoring function, employing LASER (Artetxe and Schwenk, 2019) for scoring alignment candidates. In the alignment algorithm, recursive approximation is used to reduce the search space.

### 3 The SentAlign System

In this section, we present SentAlign, a highly accurate sentence aligner capable of evaluating all possible alignment paths through fairly large documents, using a LaBSE-based scoring mechanism. Our alignment approach is of quadratic complexity,  $O(n^2)$ , and in order to handle very large files, we apply a DaC approach. When the total nodes in the alignment graph exceed a user-defined maximum, by default set to 4,000,000, the DaC-mechanism is activated in order to reduce the time complexity when aligning the documents.

The main components of the SentAlign system illustrated in Figure 3 are the scoring mechanism, the alignment or pathfinding algorithm, a DaC-module to deal with very large files, and a readjustment module to compensate for shortcomings in the scoring mechanism.

#### 3.1 Scoring

SentAlign uses LaBSE to score sentence-pair candidates. A minimum threshold score, defined by the user, is required for a sentence pair to be accepted. For each node  $[i : j]$  in the alignment graph (where

$i$  is a sentence in the source language and  $j$  is a sentence in the target language), scores for all possible alignment combinations ending in that node are calculated. The user can set a maximum number of sentences that can be merged on either side of the alignment. If merging up to three sentences on each side is allowed, a total of  $3 \times 3 = 9$  scores are compared for each node, as illustrated in Figure 4. If no alignment reaches the LaBSE threshold score,  $S_{min}$ , insertion and deletion functions are applied and the edges to the node obtain the score  $S_{min}$ . If the user wants to penalize long sentences, a user-defined maximum can be set for the number of words in either language. When either side of an alignment exceeds that maximum, a penalty is applied to the alignment score. The user can also define a maximum number of segment merges before a penalty is applied. That penalty is only applied in the pathfinding-phase (Section 3.2) and not when readjusting the path (Section 3.4). This penalty is set in order to favour shorter alignments and to deter the aligner from merging multiple sentences in one alignment when it is possible to find multiple shorter alignments instead. SentAlign seeks a maximum score for a given node in the alignment graph,  $S_{node}$ , and finds it by adding the alignment scores to the score of the node they connect from after penalties are applied.

#### 3.2 Pathfinding

The alignment problem can be seen as a way of finding the optimal path through an  $N \times M$  matrix, where  $N$  and  $M$  are the number of source and target sentences, respectively. As we allow for insertions, deletions and merges of multiple sentences on either side, we calculate the best path from the initial node  $[0, 0]$  to all other nodes in the graph using a version of Dijkstra’s algorithm (Dijkstra, 1959). Our objective is to maximize the score at each node, in contrast to the original algorithm,

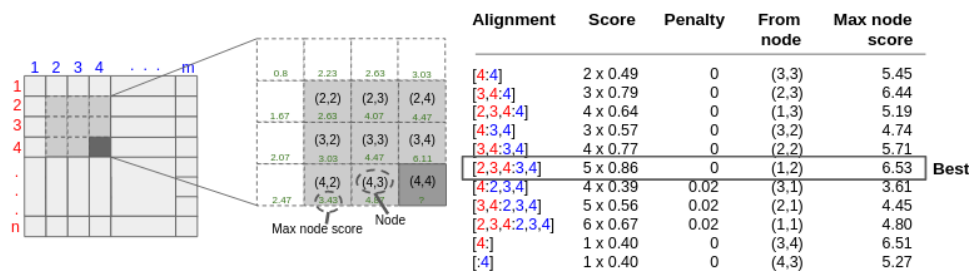


Figure 5: The maximum node score is calculated by adding the alignment score to the previously calculated maximum score of the node the alignment leads from. The LaBSE score is multiplied by the number of sentences comprising the alignment, e.g. alignment [2,3,4:3,4] has five sentences and thus the LaBSE score is multiplied by five. The max score for the node is found by adding the maximum score for node (1,2) to the alignment score.

which minimizes scores. This allows for large missing parts of text in either language without straying from the right path.

After all possible alignment scores have been calculated for a given node, an alignment function is chosen. If none of the alignments reach  $S_{min}$ , insertion and deletion alignment functions are applied and  $S_{min}$  assigned to the value of the resulting null alignments. If one or more of the possible  $n-m$  ( $n \geq 1$ ) alignments has a score above the  $S_{min}$  threshold, we assign the alignment edge a value equalling the LaBSE score multiplied by the total number of sentences merged in both languages, and add penalty-adjustments to calculate the alignment score, as illustrated in Figure 5. Finally, we select the alignment obtaining in the highest score for  $S_{node}$ . This process is repeated for each node until node  $(n, m)$  is reached. At that point, we have the optimal score from  $(0, 0)$  to  $(n, m)$  and mark the path by tracing backwards through the recorded edges.

### 3.3 Divide and Conquer

With more lines to align, the search space grows exponentially, affecting alignment speed. Zhang (2022) shows that for a quadratic time complexity sentence-alignment algorithm, chunking the parallel texts to be aligned using hard delimiters can reduce the time complexity to  $O(n \log n)$ . SentAlign allows the user to define a threshold for dividing up the search space. If the search space is larger than the user-defined threshold allows, the tool searches for high-confidence alignments to use as hard delimiters for dividing the search space into multiple smaller chunks,  $k + 1$  chunks for  $k$  hard delimiters. The aim is to find the minimum amount of alignments to use as hard delimiters to split the parallel texts into chunks of manageable size.

SentAlign looks for 1–1 alignments in the middle half of the parallel texts to use as hard delimiters, with the middle half defined as the sentences in between the first and last 25% of the sentences in the texts. One of two approaches is chosen, depending on the size of the files to align. The first choice is to employ the Gale–Church algorithm to align the parallel text/chunk under consideration, score the resulting 1–1 alignments using LaBSE and choose the highest-scoring alignment as a hard delimiter. If the parallel files are very large, running Gale–Church will take an excessive amount of time so SentAlign uses a fallback approach. When file size surpasses a second threshold, it resorts to a greedy algorithm that calculates LaBSE scores for 1–1 alignments in the allowed range and selects the highest one. When the hard-delimiter is found, the parallel text is split into two chunks. If the chunks are still too large, the process is repeated until all chunks of parallel text have the desired search space size.

### 3.4 Readjusting the Path

Thompson and Koehn (2019) argue that sentence alignment should seek a minimal parallel pair, the pair having the fewest mergers while still being truly parallel. They find that dynamic programming with cosine similarity favours many-to-many alignments over 1–1 alignments, an effect we also find when using the scoring and alignment mechanism described above. To counteract this and produce more accurate alignments, SentAlign finishes by re-evaluating each alignment in the selected path by taking another look at mergers, insertions and deletions.

First, SentAlign investigates all  $n \times m$  alignments, where  $(n > 1)$  and  $(m > 1)$ , and searches for the highest-scoring alignment which is a sub-

set of the one being investigated. If one is found that has a higher score than the original alignment, SentAlign amends the alignment path to add that as well as any other sentence pairs scoring above  $S_{min}$ . If any sentences are left they are added to the list of null alignments, containing previous insertions and deletions. Second, SentAlign looks at the list of non-aligned source and target sentences, i.e. null alignments. If a non-aligned sentence is adjacent to a sentence which has been aligned, SentAlign tries merging it to that alignment and calculates the LaBSE score. If the score increases, the path is amended. This is repeated until no more amendments can be made.

When the re-evaluation is finished, SentAlign writes out the set of alignments generated by the selected path through the alignment graph.

## 4 Evaluation

We evaluated SentAlign by comparing the system to other sentence aligners, both using sentence alignment evaluation sets and by testing the impact on downstream MT task.

### 4.1 Two evaluation sets

We compared SentAlign to five other sentence aligners: Vecalign, Bleualign, Gargantua, Hunalign and Gale-Church (using their default settings). We used two evaluation sets:

1. The manually aligned German–French evaluation set created from the Text+Berg corpus (Volk et al., 2010), first used to evaluate Bleualign and commonly used for sentence alignment evaluation since then.
2. We compiled an evaluation set for English–Icelandic sentence alignment from 10 aligned documents in five subcorpora of the ParIce corpus (Barkarson and Steingrímsson, 2019). The evaluation set (Steingrímsson, 2021) is available under an open licence and contains a total of 549 sentence alignments.<sup>2</sup> These documents are arguably easier to align than the Text+Berg documents, as none of them contain long stretches of non-alignments and there are few  $n-m$  merging alignments.

When translating the evaluation sets for Bleualign, we use OPUS-MT<sup>3</sup> (Tiedemann and Thottingal, 2020).

<sup>2</sup><http://hdl.handle.net/20.500.12537/150>

<sup>3</sup><https://opus.nlpl.eu/Opus-MT/>

| Alignment results on Text+Berg |             |             |             |             |             |             |
|--------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Algorithm                      | Strict      |             |             | Lax         |             |             |
|                                | P           | R           | $F_1$       | P           | R           | $F_1$       |
| Gargantua                      | 0.76        | 0.75        | 0.76        | 0.89        | 0.78        | 0.83        |
| Hunalign                       | 0.66        | 0.69        | 0.67        | 0.86        | 0.74        | 0.80        |
| Gale–Ch.                       | 0.68        | 0.69        | 0.69        | 0.80        | 0.73        | 0.76        |
| Vecalign                       | 0.90        | 0.90        | 0.90        | 0.99        | 0.91        | 0.95        |
| Bleualign                      | 0.93        | 0.66        | 0.77        | <b>1.00</b> | 0.68        | 0.81        |
| SentAlign                      | <b>0.94</b> | <b>0.93</b> | <b>0.93</b> | <b>1.00</b> | <b>0.93</b> | <b>0.96</b> |

Table 1: Evaluating on the German–French Text+Berg evaluation set. The highest scores are in bold. SentAlign outperforms all systems both for the strict and lax conditions, although Bleualign has a perfect score for precision, just like SentAlign.

We used the development set from the Text+Berg corpus to search for the best parameters for SentAlign. We found the best  $S_{min}$  (LaBSE) threshold to be 0.4, maximum number of words per language before applying a length penalty to be 80, and the penalty for each word exceeding that maximum to be 0.01. We performed a complete search through the alignment matrix, without chunking the search space by finding anchors as all the evaluation files were within the limits for the hard delimiters.

While none of the aligners used, with the exception of Bleualign, allow reordering of sentences in cases of possible crossing alignments, there are examples of such alignments in the Text+Berg evaluation set, which makes it impossible for other aligners to attain a perfect score. Furthermore, a few entries of null alignments are missing from the files distributed with Bleualign. To maintain consistency with previous reported scores, we did not make any changes to the evaluation set. As only some null alignments are included in the evaluation set and some are not, the results can be different based on whether a given sentence aligner returns null alignments or only useful alignments. We thus only calculated precision on non-null alignments, i.e. alignments that are true sentence pairs.

Following the original Bleualign paper, in Table 1 we report results both under the strict condition where exact matches between the gold alignment and the hypothesis are demanded, and under the lax condition where a hypothesis is true if there is an overlap with a gold alignment on both language sides. Under the lax condition, a 2–2 alignment, which is recognized as two 1–1 alignments, will yield two true positives, while it would yield two false positives under the strict condition.

We use the same settings and parameters as before for all the aligners when we evaluate on the



| Algorithm | Strict      |             |             | Lax         |             |             |
|-----------|-------------|-------------|-------------|-------------|-------------|-------------|
|           | P           | R           | $F_1$       | P           | R           | $F_1$       |
| Gargantua | 0.82        | 0.76        | 0.79        | 0.89        | 0.78        | 0.83        |
| Hunalign  | 0.72        | 0.75        | 0.73        | 0.87        | 0.78        | 0.82        |
| Gale–Ch.  | 0.78        | 0.79        | 0.79        | 0.87        | 0.81        | 0.84        |
| Vecalign  | 0.92        | 0.94        | 0.93        | 0.97        | 0.95        | 0.96        |
| Bleualign | 0.93        | 0.78        | 0.85        | 0.98        | 0.79        | 0.88        |
| SentAlign | <b>0.95</b> | <b>0.95</b> | <b>0.95</b> | <b>0.99</b> | <b>0.96</b> | <b>0.97</b> |

Table 2: Evaluating on the English–Icelandic evaluation set. The highest scores are in bold. SentAlign outperforms other systems and Vecalign is the only other aligner that comes close.

English–Icelandic evaluation set. As with the evaluation set from Text+Berg, the sentence embeddings-based alignment systems SentAlign and Vecalign are the most accurate. Using this evaluation set, the scores are higher for all aligners (see Table 2). Even though we are missing a development set for the en–is language pair and used the SentAlign parameters set for the Text+Berg de–fr development set, SentAlign does well. The results might possibly improve even more if we were to search for the best values for this sort of en–is data as the acceptance threshold for LaBSE may be different for different language pairs. While we found that 0.4 was the optimum threshold score for the Text+Berg corpus, Feng et al. (2022) set their threshold when mining sentences from CommonCrawl to 0.6. This suggests that analysis of the languages to be processed could be useful on a case-by-case basis.

## 4.2 Downstream MT

For the downstream MT task, we aligned English and Icelandic documents containing EEA regulations and directives. These documents are available as a subcorpus of ParlCe 21.10<sup>4</sup> which is published with an evaluation set in that domain.<sup>5</sup> We used fairseq (Ott et al., 2019) to train Transformer<sub>BASE</sub> models (Vaswani et al., 2017), and SacreBleu (Post, 2018) to calculate BLEU scores and statistical significance using the pairwise bootstrap test (Koehn, 2004). Table 3 reports the results for all systems, showing that SentAlign achieved the best results of the six aligners evaluated, with BLEU scores of 42.8 and 53.6, for en→is and is→en, respectively. A significance test shows that this is significantly better than all the other aligners.

<sup>4</sup><http://hdl.handle.net/20.500.12537/145>

<sup>5</sup><http://hdl.handle.net/20.500.12537/146>

| Sentence Aligner | no. pairs | en→is       | is→en       |
|------------------|-----------|-------------|-------------|
| Gargantua        | 606,768   | 39.1        | 48.9        |
| Hunalign         | 717,879   | 41.4        | 52.1        |
| Gale–Church      | 683,813   | 41.8        | 51.4        |
| Vecalign         | 670,595   | 41.8        | 51.7        |
| Bleualign        | 627,019   | 42.0        | 53.0        |
| SentAlign        | 877,485   | <b>42.8</b> | <b>53.6</b> |

Table 3: Results for MT systems trained on sentence pairs generated by different alignment tools. The table shows number of aligned pairs generated by the tools and BLEU scores for the MT systems. Bold and italic scores are the highest scores for each category and significantly higher than other systems.

## 5 Conclusion

SentAlign is an accurate, scalable and easy-to-use sentence alignment system. It uses the LaBSE model, which has been trained to generate sentence embeddings in 109 languages, to score alignment candidates. The alignment algorithm considers all possible paths through the alignment graph where the number of merges for adjoining sentences in each language is under a user-set threshold, and the maximum number of nodes in the search space is less than the DaC-threshold. Evaluation on two sentence alignment evaluation sets, as well as on a downstream MT task, show that the aligner is highly competitive, outperforming other alignment systems in most regards. SentAlign is distributed under an Apache 2.0 licence.

## Limitations

SentAlign can deliver accurate results for medium to high-resource languages in common scenarios. It is capable of evaluating all possible alignment paths through the alignment graph for parallel documents. However, as the documents get larger this may be at the cost of speed and, for very large documents, alignment time would be too long for practical use. To address this, our DaC-mechanism is applied, which enables the alignment of very large documents within reasonable time limits. Nevertheless, we can expect the system to run into problems when the number of lines in each document reaches multiple tens of thousands, due to memory constraints as well as the time factor.

LaBSE is trained on 109 languages. As noted in Section 4.1, the optimal minimum score threshold may be different between language pairs, impacting insertions and deletion made by the aligner. Furthermore, we can expect the accuracy of our

scoring function to fall if the tool is used on languages not represented in the LaBSE training data.

Finally, we used the default OPUS-MT models for aligning with Bleualign. By replacing the OPUS-MT models with higher quality models, the results for Bleualign may be further improved.

## Acknowledgements

This work was supported by the The Icelandic Centre for Research, RANNIS grant number 228654-051, and by the ADAPT Centre for Digital Content Technology which is funded under the Science Foundation Ireland (SFI) Research Centres Programme (Grant No. 13/RC/2106) and is co-funded under the European Regional Development Fund.

## References

- Mikel Artetxe and Holger Schwenk. 2019. [Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond](#). *Transactions of the Association for Computational Linguistics*, 7:597–610.
- Starkaður Barkarson and Steinþór Steingrímsson. 2019. [Compiling and Filtering ParIce: An English-Icelandic Parallel Corpus](#). In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 140–145, Turku, Finland. Linköping University Electronic Press.
- Fabienne Braune and Alexander Fraser. 2010. [Improved Unsupervised Sentence Alignment for Symmetrical and Asymmetrical Parallel Corpora](#). In *Coling 2010: Posters*, pages 81–89, Beijing, China. Coling 2010 Organizing Committee.
- Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. [A Statistical Approach to Machine Translation](#). *Computational Linguistics*, 16(2):79–85.
- Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. [Re-evaluating the Role of Bleu in Machine Translation Research](#). In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256, Trento, Italy. Association for Computational Linguistics.
- Alexis Conneau and Guillaume Lample. 2019. [Cross-lingual Language Model Pretraining](#). In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, volume 32, page 7059–7069, Vancouver, Canada. Curran Associates, Inc.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Edsger W Dijkstra. 1959. [A note on two problems in connexion with graphs](#). *Numerische mathematik*, 1(1):269–271.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. [Language-agnostic BERT Sentence Embedding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.
- William A. Gale and Kenneth W. Church. 1991. [A Program for Aligning Sentences in Bilingual Corpora](#). In *29th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, Berkeley, California. Association for Computational Linguistics.
- Huda Khayrallah and Philipp Koehn. 2018. [On the Impact of Various Types of Noise on Neural Machine Translation](#). In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.
- Philipp Koehn. 2004. [Statistical Significance Tests for Machine Translation Evaluation](#). In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.
- 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.
- 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. 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.
- Rico Sennrich and Martin Volk. 2010. [MT-based Sentence Alignment for OCR-generated Parallel Texts](#). In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas: Research Papers*, Denver, Colorado. Association for Machine Translation in the Americas.

- Rico Sennrich and Martin Volk. 2011. [Iterative, MT-based Sentence Alignment of Parallel Texts](#). In *Proceedings of the 18th Nordic Conference of Computational Linguistics*, pages 175–182, Riga, Latvia. Northern European Association for Language Technology (NEALT).
- Steinþór Steingrímsson. 2021. [Icelandic-English test set for sentence alignment 21.10](#). CLARIN-IS.
- Brian Thompson and Philipp Koehn. 2019. [Vecalign: Improved Sentence Alignment in Linear Time and Space](#). 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 1342–1348, Hong Kong, China. Association for Computational Linguistics.
- 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.
- Dániel Varga, Péter Halácsy, András Kornai, Nagy Viktor, Nagy László, Németh László, and Tron Viktor. 2005. [Parallel corpora for medium density languages](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 590–596, Borovets, Bulgaria. INCOMA Ltd.
- 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 30 (NIPS 2017)*, pages 5999–6009, Long Beach, California.
- Martin Volk, Noah Bubenhofer, Adrian Althaus, Maya Bangerter, Lenz Furrer, and Beni Ruef. 2010. [Challenges in Building a Multilingual Alpine Heritage Corpus](#). In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, pages 1653–1659, Valletta, Malta. European Language Resources Association (ELRA).
- Wu Zhang. 2022. [Improve Sentence Alignment by Divide-and-conquer](#). *ArXiv*, abs/2201.06907.

# QACHECK: A Demonstration System for Question-Guided Multi-Hop Fact-Checking

Liangming Pan<sup>1,2</sup> Xinyuan Lu<sup>3</sup> Min-Yen Kan<sup>3</sup> Preslav Nakov<sup>1</sup>

<sup>1</sup>MBZUAI <sup>2</sup>University of California, Santa Barbara <sup>3</sup>National University of Singapore

liangmingpan@ucsb.edu luxinyuan@u.nus.edu  
kanmy@comp.nus.edu.sg preslav.nakov@mbzuai.ac.ae

## Abstract

Fact-checking real-world claims often requires complex, multi-step reasoning due to the absence of direct evidence to support or refute them. However, existing fact-checking systems often lack transparency in their decision-making, making it challenging for users to comprehend their reasoning process. To address this, we propose the *Question-guided Multi-hop Fact-Checking* (QACHECK) system, which guides the model’s reasoning process by asking a series of questions critical for verifying a claim. QACHECK has five key modules: a claim verifier, a question generator, a question-answering module, a QA validator, and a reasoner. Users can input a claim into QACHECK, which then predicts its veracity and provides a comprehensive report detailing its reasoning process, guided by a sequence of (question, answer) pairs. QACHECK<sup>1</sup> also provides the source of evidence supporting each question, fostering a transparent, explainable, and user-friendly fact-checking process.

## 1 Introduction

In our age characterized by large amounts of both true and false information, fact-checking is not only crucial for counteracting misinformation but also plays a vital role in fostering trust in AI systems. However, the process of validating real-world claims is rarely straightforward. Unlike the simplicity of supporting or refuting a claim with a single piece of direct evidence, real-world claims often resemble multi-layered puzzles that require complex and multi-step reasoning to solve (Jiang et al., 2020; Nguyen et al., 2020; Aly and Vlachos, 2022; Chen et al., 2022; Pan et al., 2023).

As an example, to verify the claim “Sunlight can reach the deepest part of the Black Sea.”, it may be challenging to find direct evidence on the web that

<sup>1</sup>QACHECK is public available at <https://github.com/XinyuanLu00/QACheck>. A recorded video is at <https://www.youtube.com/watch?v=ju8kxSldM64>

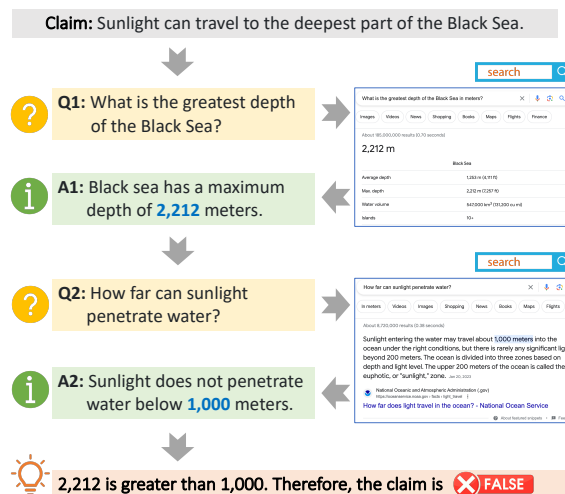


Figure 1: An example of *question-guided* reasoning for fact-checking complex real-world claims.

refutes or supports this claim. Instead, a human fact-checker needs to decompose the claim, gather multiple pieces of evidence, and perform step-by-step reasoning (Pan et al., 2023). This reasoning process can be formulated as *question-guided reasoning*, where the verification of the claim is guided by asking and answering a series of relevant questions, as shown in Figure 1. In this example, we sequentially raise two questions: “*What is the greatest depth of the Black Sea?*” and “*How far can sunlight penetrate water?*”. After independently answering these two questions by gathering relevant information from the Web, we can assert that the initial claim is *false* with simple reasoning.

While several models (Liu et al., 2020; Zhong et al., 2020; Aly and Vlachos, 2022) have been proposed to facilitate multi-step reasoning in fact-checking, they generally lack transparency in their reasoning processes. These models simply take a claim as input, then output a veracity label without an explicit explanation. Recent attempts, such as *Quin+* (Samarinas et al., 2021) and *WhatTheWiki-Fact* (Chernyavskiy et al., 2021), have aimed to develop more explainable fact-checking systems, by

searching and visualizing the supporting evidence for a given claim. However, these systems primarily validate a claim from a *single* document, and do not provide a detailed, step-by-step visualization of the reasoning process as shown in Figure 1.

We introduce the *Question-guided Multi-hop Fact-Checking* (QACHECK) system, which addresses the aforementioned issues by generating multi-step explanations via question-guided reasoning. To facilitate an explainable reasoning process, QACHECK manages the reasoning process by guiding the model to self-generate a series of questions vital for claim verification. Our system, as depicted in Figure 2, is composed of five modules: 1) a *claim verifier* that assesses whether sufficient information has been gathered to verify the claim, 2) a *question generator* to generate the next relevant question, 3) a *question-answering* module to answer the raised question, 4) a *QA validator* to evaluate the usefulness of the generated (Q, A) pair, and 5) a *reasoner* to output the final veracity label based on all collected contexts.

QACHECK offers enough adaptability, allowing users to customize the design of each module by integrating with different models. For example, we provide three alternative implementations for the QA component: the retriever-reader model, the FLAN-T5 model, and the GPT3-based reciter-reader model. Furthermore, we offer a user-friendly interface for users to fact-check any input claim and visualize its detailed question-guided reasoning process. The screenshot of our user interface is shown in Figure 4. We will discuss the implementation details of the system modules in Section 3 and some evaluation results in Section 4. Finally, we present the details of the user interface in Section 5. and conclude and discuss future work in Section 6.

## 2 Related Work

**Fact-Checking Systems.** The recent surge in automated fact-checking research aims to mitigate the spread of misinformation. Various fact-checking systems, for example, TANBIH<sup>2</sup> (Zhang et al., 2019), PRTA<sup>3</sup> (Martino et al., 2020), and WHATTHEWIKIFACT<sup>4</sup> (Chernyavskiy et al., 2021) predominantly originating from Wikipedia and claims within political or scientific domains, have facilitated this endeavor. However, the major-

<sup>2</sup><https://www.tanbih.org/about>

<sup>3</sup><https://propaganda.qcri.org/>

<sup>4</sup><https://www.tanbih.org/whatthewikifact>

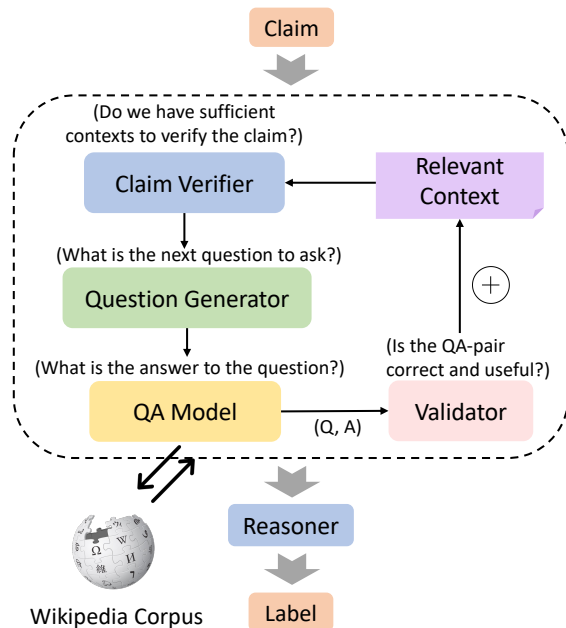


Figure 2: The architecture of our QACHECK system.

ity of these systems limit the validation or refutation of a claim to a single document, indicating a gap in systems for multi-step reasoning (Pan et al., 2023). The system most similar to ours is *Quin+* (Samarinas et al., 2021), which demonstrates evidence retrieval in a single step. In contrast, our QACHECK shows a question-led multi-step reasoning process with explanations and retrieved evidence for each reasoning step. In summary, our system 1) supports fact-checking real-world claims that require multi-step reasoning, and 2) enhances transparency and helps users have a clear understanding of the reasoning process.

**Explanation Generation.** Simply predicting a veracity label to the claim is not persuasive, and can even enhance mistaken beliefs (Guo et al., 2022). Hence, it is necessary for automated fact-checking methods to provide explanations to support model predictions. Traditional approaches have utilized attention weights, logic, or summary generation to provide post-hoc explanations for model predictions (Lu and Li, 2020; Ahmadi et al., 2019; Kotonya and Toni, 2020; Jolly et al., 2022; Xing et al., 2022). In contrast, our approach employs *question-answer* pair based explanations, offering more human-like and natural explanations.

## 3 System Architecture

Figure 2 shows the general architecture of our system, comprised of five principal modules: a Claim

Verifier  $\mathcal{D}$ , a Question Generator  $\mathcal{Q}$ , a Question-Answering Model  $\mathcal{A}$ , a Validator  $\mathcal{V}$ , and a Reasoner  $\mathcal{R}$ . We first initialize an empty context  $\mathcal{C} = \emptyset$ . Upon the receipt of a new input claim  $c$ , the system first utilizes the *claim verifier* to determine the sufficiency of the existing context to validate the claim, *i.e.*,  $\mathcal{D}(c, \mathcal{C}) \rightarrow \{\text{True}, \text{False}\}$ . If the output is False, the *question generator* learns to generate the next question that is necessary for verifying the claim, *i.e.*,  $\mathcal{Q}(c, \mathcal{C}) \rightarrow q$ . The *question-answering* model is then applied to answer the question and provide the supported evidence, *i.e.*,  $\mathcal{A}(q) \rightarrow a, e$ , where  $a$  is the predicted answer, and  $e$  is the retrieved evidence that supports the answer. Afterward, the *validator* is used to validate the usefulness of the newly-generated (Q, A) pair based on the existing context and the claim, *i.e.*,  $\mathcal{V}(c, \{q, a\}, \mathcal{C}) \rightarrow \{\text{True}, \text{False}\}$ . If the output is True, the  $(q, a)$  pair is added into the context  $\mathcal{C}$ . Otherwise, the question generator is asked to generate another question. We repeat this process of calling  $\mathcal{D} \rightarrow \mathcal{Q} \rightarrow \mathcal{A} \rightarrow \mathcal{V}$  until the claim verifier returns a True indicating that the current context  $\mathcal{C}$  contains sufficient information to verify the claim  $c$ . In this case, the *reasoner* module is called to utilize the stored relevant context to justify the veracity of the claim and outputs the final label, *i.e.*,  $\mathcal{R}(c, \mathcal{C}) \rightarrow \{\text{Supported}, \text{Refuted}\}$ . The subsequent sections provide a comprehensive description of the five key modules in QACHECK.

### 3.1 Claim Verifier

The claim verifier is a central component of QACHECK, with the specific role of determining if the current context information is sufficient to verify the claim. This module is to ensure that the system can efficiently complete the claim verification process without redundant reasoning. We build the claim verifier based on InstructGPT (Ouyang et al., 2022), utilizing its powerful *in-context learning* ability. Recent large language models such as InstructGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) have demonstrated strong few-shot generalization ability via *in-context learning*, in which the model can efficiently learn a task when prompted with the instruction of the task together with a small number of demonstrations. We take advantage of InstructGPT’s *in-context learning* ability to implement the claim verifier. We prompt InstructGPT with ten distinct *in-context* examples as detailed in Appendix A.1, where each example

consists of a claim and relevant question–answer pairs. We then prompt the model with the claim, the context, and the following instruction:

```
Claim = CLAIM
We already know the following:
CONTEXT
Can we know whether the claim is
true or false now? Yes or no?
```

If the response is ‘no’, we proceed to the question generator module. Conversely, if the response is ‘yes’, the process jumps to call the reasoner module.

### 3.2 Question Generator

The question generator module is called when the initial claim lacks the necessary context for verification. This module aims to generate the next relevant question needed for verifying the claim. Similar to the claim verifier, we also leverage InstructGPT for *in-context learning*. We use slightly different prompts for generating the initial question and the follow-up questions. The detailed prompts are in Appendix A.2. For the *initial* question generation, the instruction is:

```
Claim = CLAIM
To verify the above claim, we can
first ask a simple question:
```

For *follow-up* questions, the instruction is:

```
Claim = CLAIM
We already know the following:
CONTEXT
To verify the claim, what is the
next question we need to know the
answer to?
```

### 3.3 Question Answering Model

After generating a question, the Question Answering (QA) module retrieves corresponding evidence and provides an answer as the output. The system’s reliability largely depends on the accuracy of the QA module’s responses. Understanding the need for different QA methods in various fact-checking scenarios, we introduce three different implementations for the QA module, as shown in Figure 3.

**Retriever–Reader.** We first integrate the well-known *retriever–reader* framework, a prevalent QA paradigm originally introduced by Chen et al. (2017). In this framework, a *retriever* first retrieves relevant documents from a large evidence

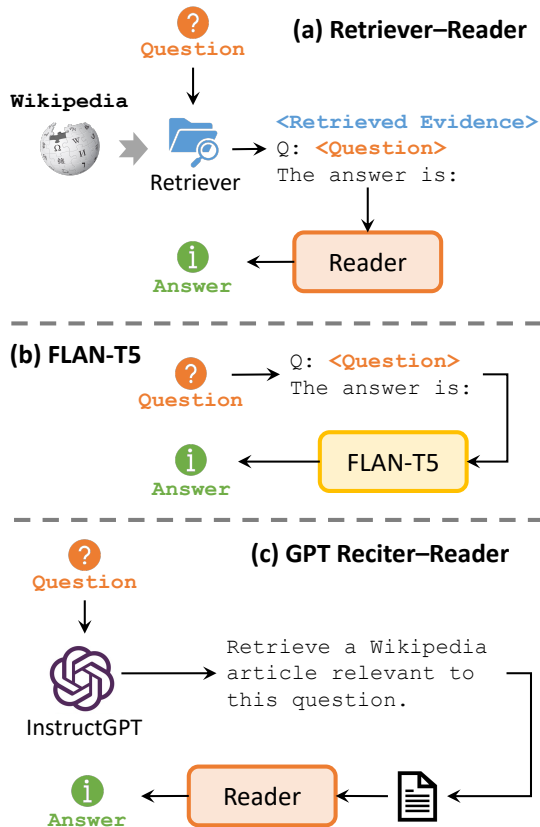


Figure 3: Illustrations of the three different implementations of the Question Answering module in QACHECK.

corpus, and then a *reader* predicts an answer conditioned on the retrieved documents. For the evidence corpus, we use the Wikipedia dump provided by the Knowledge-Intensive Language Tasks (KILT) benchmark (Petroni et al., 2021), in which the Wikipedia articles have been pre-processed and separated into paragraphs. For the retriever, we apply the widely-used sparse retrieval based on BM25 (Robertson and Zaragoza, 2009), implemented with the Pyserini toolkit (Lin et al., 2021). For the reader, we use the *RoBERTa-large* (Liu et al., 2019) model fine-tuned on the SQuAD dataset (Rajpurkar et al., 2016), using the implementation from *PrimeQA*<sup>5</sup> (Sil et al., 2023).

**FLAN-T5.** While effective, the retriever-reader framework is constrained by its reliance on the evidence corpus. In scenarios where a user’s claim is outside the scope of Wikipedia, the system might fail to produce a credible response. To enhance flexibility, we also incorporate the *FLAN-T5* model (Chung et al., 2022), a Seq2Seq model pre-trained on more than 1.8K tasks with instruction

tuning. It directly takes the question as input and then generates the answer and the evidence, based on the model’s parametric knowledge.

**GPT Reciter-Reader.** Recent studies (Sun et al., 2023; Yu et al., 2023) have demonstrated the great potential of the GPT series, such as InstructGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023), to function as robust knowledge repositories. The knowledge can be retrieved by properly prompting the model. Drawing from this insight, we introduce the *GPT Reciter-Reader* approach. Given a question, we prompt the InstructGPT to “recite” the knowledge stored within it, and InstructGPT responds with relevant evidence. The evidence is then fed into a *reader* model to produce the corresponding answer. While this method, like FLAN-T5, does not rely on a specific corpus, it stands out by using InstructGPT. This offers a more dependable parametric knowledge base than FLAN-T5.

The above three methods provide a flexible and robust QA module, allowing for switching between the methods as required, depending on the claim being verified and the available contextual information. In the following, we use GPT Reciter-Reader as the default implementation for our QA module.

### 3.4 QA Validator

The validator module ensures the usefulness of the newly-generated QA pairs. For a QA pair to be valid, it must satisfy two criteria: 1) it brings additional information to the current context  $\mathcal{C}$ , and 2) it is useful for verifying the original claim. We again implement the validator by prompting InstructGPT with a suite of ten demonstrations shown in Appendix A.3. The instruction is as follows:

```
Claim = CLAIM
We already know the following:
CONTEXT
Now we further know:
NEW QA PAIR
Does the QA pair have additional
knowledge useful for verifying
the claim?
```

The validator acts as a safeguard against the system producing redundant or irrelevant QA pairs. Upon validation of a QA pair, it is added to the current context  $\mathcal{C}$ . Subsequently, the system initiates another cycle of calling the claim verifier, question generator, question answering, and validation.

<sup>5</sup><https://github.com/primeqa/primeqa>



Figure 4: The screenshot of the QACHECK user interface showing its key annotated functions. First, users have the option to *select a claim* or *manually input* a claim that requires verification. Second, users can start the verification process by clicking the *Submit* button. Third, the system shows a step-by-step question-answering guided reasoning process. Each step includes the *reasoning depth*, the *generated question*, relevant retrieved *evidence*, and the corresponding predicted *answer*. Finally, it presents the final prediction *label* with the supporting *rationale*.

### 3.5 Reasoner

The reasoner is called when the claim verifier determines that the context  $\mathcal{C}$  is sufficient to verify the claim or the system hits the maximum allowed iterations, set to 5 by default. The reasoner is a special question-answering model which takes the context  $\mathcal{C}$  and the claim  $c$  as inputs and then answers the question “*Is the claim true or false?*”. The model is also requested to output the rationale with the prediction. We provide two different implementa-

tions for the reasoner: 1) the end-to-end QA model based on FLAN-T5, and 2) the InstructGPT model with the prompts given in Appendix A.4.

## 4 Performance Evaluation

To evaluate the performance of our QACHECK, we use two fact-checking datasets that contain complex claims requiring multi-step reasoning: HOVER (Jiang et al., 2020) and FEVEROUS (Aly et al., 2021), following the same experimental set-



| Model       | HOVER        |              |              | FEVEROUS     |
|-------------|--------------|--------------|--------------|--------------|
|             | 2-hop        | 3-hop        | 4-hop        |              |
| InstructGPT |              |              |              |              |
| - Direct    | 56.51        | 51.75        | 49.68        | 60.13        |
| - CoT       | <b>57.20</b> | 53.66        | 51.83        | <b>61.05</b> |
| Codex       | 55.57        | 53.42        | 45.59        | 57.85        |
| FLAN-T5     | 48.27        | 52.11        | 51.13        | 55.16        |
| ProgramFC   | 54.27        | 54.18        | <b>52.88</b> | 59.66        |
| QACheck     | 55.67        | <b>54.67</b> | 52.35        | 59.47        |

Table 1: Evaluation of F1 scores for different models. The bold text shows the best results for each setting.

tings used in Pan et al. (2023). HOVER contains 1,126 two-hop claims, 1,835 three-hop claims, and 1,039 four-hop claims, while FEVEROUS has 2,962 multi-hop claims. We compare our method with the baselines of directly applying InstructGPT with two different prompting methods: (i) *direct* prompting with the claim, and (ii) CoT (Wei et al., 2022) or chain-of-thought prompting with few-shot demonstrations of reasoning explanations. We also compare with ProgramFC (Pan et al., 2023), FLAN-T5 (Chung et al., 2022), and Codex (Chen et al., 2021). We use the reported results for the baseline models from Pan et al. (2023).

The evaluation results are shown in Table 1. Our QACHECK system achieves a macro-F1 score of 55.67, 54.67, and 52.35 on HOVER two-hop, three-hop, and four-hop claims, respectively. It achieves a 59.47 F1 score on FEVEROUS. These scores are better than directly using InstructGPT, Codex, or FLAN-T5. They are also on par with the systems that apply claim decomposition strategies, *i.e.*, CoT, and ProgramFC. The results demonstrate the effectiveness of our QACHECK system. Especially, the QACHECK has better improvement over the end-to-end models on claims with high reasoning depth. This indicates that decomposing a complex claim into simpler steps with question-guided reasoning can facilitate more accurate reasoning.

## 5 User Interface

We create a demo system based on Flask<sup>6</sup> for verifying open-domain claims with QACHECK, as shown in Figure 4. The QACHECK demo is designed to be intuitive and user-friendly, enabling users to input any claim or select from a list of pre-defined claims (top half of Figure 4).

<sup>6</sup><https://flask.palletsprojects.com/en/2.3.x/>

Upon selecting or inputting a claim, the user can start the fact-checking process by clicking the “Submit” button. The bottom half of Figure 4 shows a snapshot of QACHECK’s output for the input claim “Lars Onsager won the Nobel prize when he was 30 years old”. The system visualizes the detailed question-guided reasoning process. For each reasoning step, the system shows the index of the reasoning step, the generated question, and the predicted answer to the question. The retrieved evidence to support the answer is shown on the right for each step. The system then shows the final veracity prediction for the original claim accompanied by a comprehensive rationale in the “Prediction with rationale” section. This step-by-step illustration not only enhances the understanding of our system’s fact-checking process but also offers transparency to its functioning.

QACHECK also allows users to change the underlying question-answering model. As shown at the top of Figure 4, users can select between the three different QA models introduced in Section 3.3, depending on their specific requirements or preferences. Our demo system will be open-sourced under the Apache-2.0 license.

## 6 Conclusion and Future Works

This paper presents the QACHECK system, a novel approach designed for verifying real-world complex claims. QACHECK conducts the reasoning process with the guidance of asking and answering a series of questions and answers. Specifically, QACHECK iteratively generates contextually relevant questions, retrieves and validates answers, judges the sufficiency of the context information, and finally, reasons out the claim’s truth value based on the accumulated knowledge. QACHECK leverages a wide range of techniques, such as in-context learning, document retrieval, and question-answering, to ensure a precise, transparent, explainable, and user-friendly fact-checking process.

In the future, we plan to enhance QACHECK 1) by integrating additional knowledge bases to further improve the breadth and depth of information accessible to the system (Feng et al., 2023; Kim et al., 2023), and 2) by incorporating a multi-modal interface to support image (Chakraborty et al., 2023), table (Chen et al., 2020; Lu et al., 2023), and chart-based fact-checking (Akhtar et al., 2023), which can broaden the system’s utility in processing and analyzing different forms of data.

## Limitations

We identify two main limitations of QACHECK. First, several modules of our QACHECK currently utilize external API-based large language models, such as InstructGPT. This reliance on external APIs tends to prolong the response time of our system. As a remedy, we are considering the integration of open-source, locally-run large language models like LLaMA (Touvron et al., 2023). Secondly, the current scope of our QACHECK is confined to evaluating *True/False* claims. Recognizing the significance of also addressing *Not Enough Information* claims, we plan to devise strategies to incorporate these in upcoming versions of the system.

## Ethics Statement

The use of large language models requires a significant amount of energy for computation for training, which contributes to global warming. Our work performs few-shot in-context learning instead of training models from scratch, so the energy footprint of our work is less. The large language model (InstructGPT) whose API we use for inference consumes significant energy.

## Acknowledgement

This project is supported by the Ministry of Education, Singapore, under its MOE AcRF TIER 3 Grant (MOE-MOET32022-0001). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore.

## References

- Naser Ahmadi, Joohyung Lee, Paolo Papotti, and Mohammed Saeed. 2019. [Explainable fact checking with probabilistic answer set programming](#). In *Proceedings of the 2019 Truth and Trust Online Conference (TTO)*.
- Mubashara Akhtar, Oana Cocarascu, and Elena Simperl. 2023. [Reading and reasoning over chart images for evidence-based automated fact-checking](#). In *Findings of the Association for Computational Linguistics (EACL)*, pages 399–414.
- Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. [FEVEROUS: Fact Extraction and VERification Over Unstructured and Structured information](#). In *Proceedings of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*.
- Rami Aly and Andreas Vlachos. 2022. [Natural logic-guided autoregressive multi-hop document retrieval for fact verification](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6123–6135.
- Megha Chakraborty, Khusbu Pahwa, Anku Rani, Adarsh Mahor, Aditya Pakala, Arghya Sarkar, Harshit Dave, Ishan Paul, Janvita Reddy, Preethi Gurumurthy, Ritvik G, Samahriti Mukherjee, Shreyas Chatterjee, Kinjal Sensharma, Dwip Dalal, Suryavardan S, Shreyash Mishra, Parth Patwa, Aman Chadha, Amit P. Sheth, and Amitava Das. 2023. [FACTIFY3M: A benchmark for multimodal fact verification with explainability through 5w question-answering](#). *CoRR*, abs/2306.05523.
- 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 (ACL)*, pages 1870–1879.
- Jifan Chen, Aniruddh Sriram, Eunsol Choi, and Greg Durrett. 2022. [Generating literal and implied sub-questions to fact-check complex claims](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3495–3516.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebguss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *ArXiv preprint*, abs/2107.03374.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. [Tabfact: A large-scale dataset for table-based fact verification](#). In *Proceedings of 8th International Conference on Learning Representations (ICLR)*.
- Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2021. [Whatthewikifact: Fact-checking claims against wikipedia](#). In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 4690–4695.

- 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.
- Shangbin Feng, Vidhisha Balachandran, Yuyang Bai, and Yulia Tsvetkov. 2023. [Factkb: Generalizable factuality evaluation using language models enhanced with factual knowledge](#). *CoRR*, abs/2305.08281.
- Zhijiang Guo, Michael Sejr Schlichtkrull, and Andreas Vlachos. 2022. [A survey on automated fact-checking](#). *Transactions of the Association for Computational Linguistics (TACL)*, 10:178–206.
- Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020. [HoVer: A dataset for many-hop fact extraction and claim verification](#). In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 3441–3460.
- Shailza Jolly, Pepa Atanasova, and Isabelle Augenstein. 2022. [Generating fluent fact checking explanations with unsupervised post-editing](#). *Information*, 13:500.
- Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. [Factkg: Fact verification via reasoning on knowledge graphs](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16190–16206.
- Neema Kotonya and Francesca Toni. 2020. [Explainable automated fact-checking for public health claims](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754.
- Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Frassetto Nogueira. 2021. [Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations](#). In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2356–2362.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. [Fine-grained fact verification with kernel graph attention network](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7342–7351.
- Xinyuan Lu, Liangming Pan, Qian Liu, Preslav Nakov, and Min-Yen Kan. 2023. [SCITAB: A challenging benchmark for compositional reasoning and claim verification on scientific tables](#). *CoRR*, abs/2305.13186.
- Yi-Ju Lu and Cheng-Te Li. 2020. [GCAN: graph-aware co-attention networks for explainable fake news detection on social media](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 505–514.
- Giovanni Da San Martino, Shaden Shaar, Yifan Zhang, Seunghak Yu, Alberto Barrón-Cedeño, and Preslav Nakov. 2020. [Prta: A system to support the analysis of propaganda techniques in the news](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 287–293.
- Van-Hoang Nguyen, Kazunari Sugiyama, Preslav Nakov, and Min-Yen Kan. 2020. [FANG: leveraging social context for fake news detection using graph representation](#). In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1165–1174.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- 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 *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*.
- Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023. [Fact-checking complex claims with program-guided reasoning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6981–7004.
- Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. 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 (NAACL-HLT)*, pages 2523–2544.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [Squad: 100, 000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

- Stephen E. Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: BM25 and beyond](#). *Journal of Foundations and Trends in Information Retrieval*, 3(4):333–389.
- Chris Samarinas, Wynne Hsu, and Mong-Li Lee. 2021. [Improving evidence retrieval for automated explainable fact-checking](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations (NAACL-HLT)*, pages 84–91.
- Avi Sil, Jaydeep Sen, Bhavani Iyer, Martin Franz, Kshittij Fadnis, Mihaela Bornea, Sara Rosenthal, J. Scott McCarley, Rong Zhang, Vishwajeet Kumar, Yulong Li, Md. Arafat Sultan, Riyaz Bhat, Jürgen Broß, Radu Florian, and Salim Roukos. 2023. [Primeqa: The prime repository for state-of-the-art multilingual question answering research and development](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 51–62.
- Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2023. [Recitation-augmented language models](#). In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#). *CoRR*, abs/2302.13971.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). *CoRR*, abs/2201.11903.
- Rui Xing, Shraey Bhatia, Timothy Baldwin, and Jey Han Lau. 2022. [Automatic explanation generation for climate science claims](#). In *Proceedings of the 20th Annual Workshop of the Australasian Language Technology Association (ALTA)*, pages 122–129.
- Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. [Generate rather than retrieve: Large language models are strong context generators](#). In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.
- Yifan Zhang, Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Jisun An, Haewoon Kwak, Todor Staykovski, Israa Jaradat, Georgi Karadzhov, Ramy Baly, Kareem Darwish, James R. Glass, and Preslav Nakov. 2019. [Tanbih: Get to know what you are reading](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing: System Demonstrations (EMNLP-IJCNLP)*, pages 223–228.
- Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. [Reasoning over semantic-level graph for fact checking](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6170–6180.

## A Prompts

### A.1 Prompts for Claim Verifier

```
Claim = Superdrag and Collective Soul are
both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
Can we know whether the claim is
true or false now? Yes or no?
Prediction = No, we cannot know.

Claim = Superdrag and Collective Soul are
both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
Question 2 = Is Collective Soul a rock band?
Answer 2 = Yes
Can we know whether the claim is
true or false now? Yes or no?
Prediction = Yes, we can know.

<10 demonstrations in total>
-----
Claim = [[CLAIM]]
Claim = CLAIM
We already know the following:
[[QA_CONTEXTS]]
Can we know whether the claim is
true or false now? Yes or no?
Prediction =
```

### A.2 Prompts for Question Generation

#### Prompts for the initial question generation

```
Claim = Superdrag and Collective Soul are
both rock bands.
To verify the above claim, we can
first ask a simple question:
Question = Is Superdrag a rock band?

<10 demonstrations in total>
-----
Claim = [[CLAIM]]
To verify the above claim, we can
first ask a simple question:
Question =
```

#### Prompts for the follow-up question generation

```
Claim = Superdrag and Collective Soul are
both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
To verify the claim, what is the
next question we need to know the
answer to?
Question 2 = Is Collective Soul a rock band?

<10 demonstrations in total>
-----
Claim = [[CLAIM]]
We already know the following:
[[QA_CONTEXTS]]
To verify the claim, what is the
next question we need to know the
answer to?
Question [[Q_INDEX]] =
```

### A.3 Prompts for Validator

```
Claim = Superdrag and Collective Soul are
both rock bands.
We already know the following:
Question = Is Superdrag a rock band?
Answer = Yes
Now we further know:
```

```
Question = Is Collective Soul a rock band?
Answer = Yes
Does the QA pair have additional
knowledge useful for verifying the claim?
The answer: Yes
```

```
<10 demonstrations in total>
-----
Claim = [[CLAIM]]
We already know the following:
[[QA_CONTEXTS]]
Now we further know:
[[NEW_QA_PAIR]]
Does the QA pair have additional
knowledge useful for verifying the claim?
The answer:
```

### A.4 Prompts for Reasoner

```
Contexts:
Q1: When Lars Onsager won the Nobel Prize?
A1: 1968
Q2: When was Lars Onsager born?
A2: 1903
Claim = Lars Onsager won the Nobel Prize
when he was 30 years old.

Is this claim true or false?

Answer:
Lars Onsager won the Nobel Prize in 1968.
Lars Onsager was born in 1903.
Therefore, the final answer is: False.
```

```
<10 demonstrations in total>
-----
Contexts:
[[CONTEXTS]]
Claim = [[CLAIM]]
Is this claim true or false?
Answer:
Therefore, the final answer is
```

# RobustQA: A Framework for Adversarial Text Generation Analysis on Question Answering Systems

Yasaman Boreshban<sup>1\*</sup>, Seyed Morteza Mirbostani<sup>2\*</sup>,  
Seyedeh Fatemeh Ahmadi<sup>2</sup>, Gita Shojaee<sup>2</sup>, Fatemeh Kamani<sup>2</sup>,  
Gholamreza Ghassem-Sani<sup>1</sup>, and Seyed Abolghasem Mirroshandel<sup>2</sup>

<sup>1</sup>Computer Engineering Department, Sharif University of Technology, Tehran, Iran

<sup>2</sup>Computer Engineering Department, University of Guilan, Rasht, Iran

{yasaman.boreshban, sani}@sharif.edu

m.mirbostani@msc.guilan.ac.ir mirroshandel@guilan.ac.ir

## Abstract

Question answering (QA) systems have reached human-level accuracy; however, these systems are not robust enough and are vulnerable to adversarial examples. Recently, adversarial attacks have been widely investigated in text classification. However, there have been few research efforts on this topic in QA. In this article, we have modified the attack algorithms widely used in text classification to fit those algorithms for QA systems. We have evaluated the impact of various attack methods on QA systems at character, word, and sentence levels. Furthermore, we have developed a new framework, named RobustQA, as the first open-source toolkit for investigating textual adversarial attacks in QA systems. RobustQA consists of seven modules: Tokenizer, Victim Model, Goals, Metrics, Attacker, Attack Selector, and Evaluator. It currently supports six different attack algorithms. Furthermore, the framework simplifies the development of new attack algorithms in QA.

## 1 Introduction

With the release of large and high-quality datasets in the field of question answering (QA) (Rajpurkar et al., 2016; Joshi et al., 2017; Trischler et al., 2017; Kočiský et al., 2018), we witness significant progress in this area of research. With the aid of deep neural networks (DNNs), the newly presented models have even reached human-level accuracy. However, it has been shown that these models are not yet robust enough and are vulnerable to adversarial examples (Gil et al., 2019; Ren et al., 2019).

In the context of QA systems, the model’s accuracy drops drastically when some adversarial sentences are added to the input paragraphs (Jia and Liang, 2017). Accordingly, extensive research efforts have been conducted addressing various techniques to increase the robustness of DNN models in

different fields. One of the most popular techniques to overcome this issue is the so-called adversarial training. In adversarial training, some adversarial examples are used during the training phase of the model to increase its robustness against textual adversarial attacks (Jia and Liang, 2017; Gan and Ng, 2019). In another research, the impact of the knowledge distillation technique on the robustness of QA models has been analyzed (Boreshban et al., 2023).

Adversarial attacks have been widely investigated in the field of text classification (Li et al., 2020; Jin et al., 2020). Furthermore, OpenAttack (Zeng et al., 2021) and TextAttack (Morris et al., 2020) frameworks have been presented to simplify the implementation and analysis of different attack methods in text classification. However, there has been only a limited number of efforts in this regard for the QA systems.

The contributions of the paper can be summarized as follows: 1) We modify the attack algorithms that have been widely used in the field of text classification for QA systems. 2) We show that these modified attack algorithms can easily be evaluated on QA systems in three different characters, words, and sentence levels. 3) We build a new open-source framework named RobustQA, aiming at simplifying the research on textual adversarial attacks in QA systems. 4) We have incorporated both adversarial text generation and data augmentation in RobustQA for being used in adversarial training methods to improve the robustness and generalization of QA models.

In this paper, we introduce the related works in Section 2. We compare the QA task against text classification and describe a sample textual adversarial attack algorithm implemented for the QA task in Section 3. Next, we introduce the RobustQA framework modules in detail in Section 4 and demonstrate the framework’s usage in Section 5. We present our setup and experimental

\*These authors contributed equally to this work.

results in Section 6. Finally, our conclusions and future works are presented in Section 7.

## 2 Related works

### 2.1 Adversarial Sentences in Text Classification

Adversarial attacks have been extensively studied on continuous data (Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2017); however, addressing these attacks on discrete data such as text (Xu et al., 2020; Zhang et al., 2020) poses significant challenges.

Adversarial attacks can be categorized based on different aspects. Attacks are primarily divided into two types of white and black boxes. In white box attacks, the attacker has full access to the model and its parameters. In this type of attack, the gradient of the cost function relative to the input is used to generate an adversarial example (Papernot et al., 2016; Ebrahimi et al., 2018; Li et al., 2018; Wallace et al., 2019). In the black box attack, however, there is limited knowledge regarding the model, and thus one can only use the output of the model to generate an adversarial example (Jin et al., 2020; Garg and Ramakrishnan, 2020; Li et al., 2020).

Adversarial attacks are also divided into untargeted and targeted categories. In untargeted attacks, the goal is merely to cause the model to produce an incorrect output label (Pruthi et al., 2019; Garg and Ramakrishnan, 2020); whereas in targeted ones, more restrictions are applied to impose a specific wrong prediction (Gao et al., 2018; Wang et al., 2020).

Textual adversarial attacks are divided into three categories in terms of perturbation levels, i.e., character, word, and sentence. Character-level attacks usually manipulate characters based on insertion, deletion, swap, substitution, and repetition operations (Gao et al., 2018; Eger et al., 2019; Gil et al., 2019; He et al., 2021). In word-level attacks, words are replaced with their synonyms. In this case, the algorithms consist of two stages. At first, resources such as the word embedding (Jin et al., 2020), language models (Li et al., 2020; Garg and Ramakrishnan, 2020), and semantic networks (Ren et al., 2019) are used to produce a set of perturbations. In the second stage, using different algorithms such as greedy search (Li et al., 2018; Ren et al., 2019), beam search (Ebrahimi et al., 2018), genetic algorithm (Alzantot et al., 2018), and particle swarm optimization (Zang et al., 2020), suc-

cessful queries are selected. Finally, in sentence-level attacks, special techniques such as adding misleading sentences to the text (Jia and Liang, 2017), paraphrasing (Iyyer et al., 2018; Ribeiro et al., 2018; Gan and Ng, 2019; Huang and Chang, 2021), and using the autoencoder structure (Zhao et al., 2017; Wang et al., 2020) are employed to produce adversarial sentences.

### 2.2 Adversarial Sentences in QA Systems

There have been only a few research initiatives focused on textual adversarial attacks in the field of QA.

Jia and Liang (2017) showed that QA systems get confused by appending misleading sentences to the input paragraph. They introduced two algorithms called *AddSent* and *AddAny*. Later, Yang et al. (2021) improved these algorithms by introducing *AddSentDivers* to increase the diversity of the generated adversarial sentences.

It has been demonstrated that paraphrasing the questions is an alternative method for generating adversarial sentences. In this regard, Ribeiro et al. (2018) used the back translation technique to obtain question paraphrase rules. Also, Gan and Ng (2019) used a transformer model to produce paraphrased questions and introduced two types of adversarial questions. The autoencoder structure was utilized in another recent research to generate adversarial sentences (Wang et al., 2020).

### 2.3 Available Frameworks

There are several open-source libraries for building adversarial examples on continuous data. The most notable ones are CleverHans (Papernot et al., 2016), Foolbox (Rauber et al., 2017), Adversarial Robustness Toolbox (Nicolae et al., 2018), and AsvBox (Goodman et al., 2020). On the contrary, a limited number of open-source libraries operable on the discrete data are available. SeqAttack (Simoncini and Spanakis, 2021) is a framework for conducting adversarial attacks on the named entity recognition problems. The most famous frameworks for creating adversarial sentences in text classification are OpenAttack (Zeng et al., 2021) and TextAttack (Morris et al., 2020). Although these frameworks are suitable for text classification, these algorithms do not currently support QA systems.

### 3 Question Answering vs. Text Classification

#### 3.1 Task Structure

In QA systems, the question and context are represented as a sequence of tokens,  $\mathbf{Q} = \{q_1, q_2, q_3, \dots, q_n\}$  and  $\mathbf{C} = \{c_1, c_2, c_3, \dots, c_n\}$ , respectively. In these systems, the main goal is to predict the answer,  $\mathbf{A}$ , in the form of a span within the context,  $\mathbf{A} = \{c_j, \dots, c_{j+k}\}$ . The returned span includes a specific start and an end token indices of the context paragraphs. F1 score and exact match (EM) criteria are the two common metrics for evaluating QA systems.

On the other hand, in the text classification task, the main goal is to recognize the correct class of an input text. Due to the substantial differences between QA and text classification tasks, the algorithms designed for dealing with the attacks in the text classification are not directly applicable to the QA problems. The main distinctions are related to their differences in the structure of the input data and the goal function of attack scenarios.

#### 3.2 Input Data Structure

In text classification of an input text  $\mathbf{X}$  with the corresponding ground truth label  $Y$  and the victim model  $F$ , the goal of an attack scenario is to have an attack set up that transforms  $\mathbf{X}$  to  $\tilde{\mathbf{X}}$  with the minimum perturbation in such a way that the victim model predicts an incorrect label  $\tilde{Y}$ , where  $Y \neq \tilde{Y}$ .

In QA tasks, every input  $\mathbf{X}$  is composed of a question  $\mathbf{Q}$  and a context  $\mathbf{C}$ .

$$\mathbf{X} = x_1 x_2 \cdots x_i \cdots x_n, \quad x_i \in \{\mathbf{Q}, \mathbf{C}\} \quad (1)$$

The ground truth label  $\mathbf{Y}$ , which is a part of the given context with specific start and end tokens, represents the correct answer to the given question.

$$\mathbf{Y} = c_j \cdots c_{j+k} \quad c_j \in \{\mathbf{C}\} \quad (2)$$

The predicted answer  $\tilde{\mathbf{Y}}$  is computed by considering the maximum probability for the start and end tokens.

$$\tilde{\mathbf{Y}} = F(\arg \max_{x \in \mathbf{C}} P(x)) \quad (3)$$

Akin to the text classification, the goal of an attack scenario here is to have an attack set up that transforms  $\mathbf{X}$  to  $\tilde{\mathbf{X}}$  with the minimum perturbation  $\Delta x$  in a way that the victim model predicts an incorrect answer span  $\tilde{\mathbf{Y}}$ , where

$$\tilde{x} = x + \Delta x, \quad \|\Delta x\|_p < \epsilon \quad (4)$$

$$\tilde{\mathbf{Y}} \neq \mathbf{Y} \quad (5)$$

#### 3.3 Goal Function Criteria

In both text classification and QA tasks, the goal function of an attack scenario determines the success of the attack on a given victim model. In text classification, an attack scenario for a given input example is regarded as successful if the model prediction for the example is not equal to its corresponding ground truth label. In this task, the goal function can be simply evaluated by a single criterion.

However, in QA tasks, since a prediction label includes two items (i.e., the start and the end tokens of the predicted answer span), the goal function is usually evaluated by both F1 and EM criteria.

#### 3.4 Attack Methods

To demonstrate the required modifications of an attack method to cope with the mentioned differences, we discuss the details of the changes applied to the TextFooler algorithm (Jin et al., 2020). We have applied similar changes to few other attack algorithms in the new framework to make those algorithms fit for QA tasks.

The TextFooler algorithm is a score-based textual adversarial attack that consists of two primary steps. The first step is the *word importance ranking*, in which words are sorted according to their importance. The second step is the *word transformation*, which produces suitable substitutes for the words with the highest importance level obtained from the first step to generate an adversary example.

Algorithm 1 shows the pseudo-code of a revised version of the TextFooler algorithm, which is compatible with QA tasks.

**Word Importance Ranking (line 1-11)** The input example  $\mathbf{X}$ , which includes context  $\mathbf{C}$  and question  $\mathbf{Q}$ , accompanied by its corresponding ground truth label  $\mathbf{Y}$ , is passed to the algorithm. The goal is to confuse the victim model by generating a new question  $\tilde{\mathbf{Q}}$ , with the minimum perturbation to  $\mathbf{Q}$ . One metric among the F1 score and EM measure is used for marking an adversarial example. In Algorithm 1, we use the F1 score and  $\delta$ , a threshold value empirically set to 0.9, as the goal function criterion.



---

**Algorithm 1:** QA Adversarial Attack by TextFooler

---

**Input:** Input example  $\mathbf{X} = \{\mathbf{Q}, \mathbf{C}\} = \{w_1, w_2, \dots, w_n\}$ , the corresponding ground truth label  $\mathbf{Y}$ , victim model  $F$ , victim model's prediction  $P$ , sentence similarity function  $\text{Sim}(\cdot)$ , sentence similarity threshold  $\epsilon$ , word embeddings  $\text{Emb}$  over the vocabulary  $\text{Vocab}$ , F1 score function  $F1(\cdot)$ , goal's F1 score threshold  $\delta$ , adversarial question  $\tilde{\mathbf{Q}}$

**Output:** Adversarial example  $\tilde{\mathbf{X}} = \{\tilde{\mathbf{Q}}, \mathbf{C}\}$

- 1 Initialization:  $\tilde{\mathbf{X}} \leftarrow \mathbf{X}$
- 2 **for** each word  $w_i$  in  $\mathbf{Q}$  **do**
- 3     Compute the importance score of the start and end answer span,  $I_{w_i} = (I_{w_i}^s, I_{w_i}^e)$
- 4     **if**  $F(\mathbf{X}) = F(\mathbf{X}_{\setminus w_i}) = \mathbf{Y}$  **then**
- 5          $(I_{w_i}^s, I_{w_i}^e) \leftarrow P_{\mathbf{Y}}(\mathbf{X}) - P_{\mathbf{Y}}(\mathbf{X}_{\setminus w_i})$
- 6     **else if**  $F(\mathbf{X}) = \mathbf{Y}$ ,  $F(\mathbf{X}_{\setminus w_i}) = \tilde{\mathbf{Y}}$ , and  $\mathbf{Y} \neq \tilde{\mathbf{Y}}$  **then**
- 7          $(I_{w_i}^s, I_{w_i}^e) \leftarrow (P_{\mathbf{Y}}(\mathbf{X}) - P_{\mathbf{Y}}(\mathbf{X}_{\setminus w_i})) + (P_{\tilde{\mathbf{Y}}}(\mathbf{X}_{\setminus w_i}) - P_{\tilde{\mathbf{Y}}}(\mathbf{X}))$
- 8     **end**
- 9 **end**
- 10 Create a set  $\mathbf{W}$  of all words  $w_i \in \mathbf{Q}$  sorted by the descending order of their importance score, either using start  $I_{w_i}^s$  or average  $(I_{w_i}^s + I_{w_i}^e)/2$  importance score.
- 11 Filter out the stop words in  $\mathbf{W}$ .
- 12 **for** each word  $w_j$  in  $\mathbf{W}$  **do**
- 13     Initiate the set of candidates  $\text{CANDIDATES}$  by extracting the top  $N$  synonyms using  $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$  for each word in  $\text{Vocab}$ .
- 14      $\text{CANDIDATES} \leftarrow \text{POSFilter}(\text{CANDIDATES})$
- 15      $\text{FINCANDIDATES} \leftarrow \{\}$
- 16     **for**  $c_k$  in  $\text{CANDIDATES}$  **do**
- 17          $\mathbf{X}' \leftarrow \text{Replace } w_j \text{ with } c_k \text{ in } \tilde{\mathbf{X}}$
- 18         **if**  $\text{Sim}(\mathbf{Q}', \tilde{\mathbf{Q}}) > \epsilon$  **then**
- 19              $\text{FINCANDIDATES} \leftarrow \text{FINCANDIDATES} \cup \{c_k\}$
- 20              $\mathbf{Y}_k \leftarrow F(\mathbf{X}')$
- 21              $\mathbf{P}_k \leftarrow F_{\mathbf{Y}_k}(\mathbf{X}')$
- 22         **end**
- 23     **end**
- 24      $\alpha = (F1(\mathbf{X}) - F1(\mathbf{X}'))/F1(\mathbf{X})$
- 25     **if** there exists  $c_k$  where  $\alpha > \delta$  **then**
- 26         In  $\text{FINCANDIDATES}$ , only keep the candidates  $c_k$  where  $\alpha > \delta$
- 27          $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(\mathbf{Q}, \mathbf{Q}'_{w_j \rightarrow c})$
- 28          $\tilde{\mathbf{Q}} \leftarrow \text{Replace } w_j \text{ with } c^* \text{ in } \tilde{\mathbf{Q}}$
- 29         **return**  $\{\tilde{\mathbf{Q}}, \mathbf{C}\}$
- 30     **else if**  $\min_{c_k \in \text{FINCANDIDATES}} \mathbf{P}_k < P_{\mathbf{Y}_k}(\tilde{\mathbf{X}})$  **then**
- 31          $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} \mathbf{P}_k$
- 32          $\tilde{\mathbf{Q}} \leftarrow \text{Replace } w_j \text{ with } c^* \text{ in } \tilde{\mathbf{Q}}$
- 33     **end**
- 34 **end**
- 35 **return** None

---

First, a copy of  $\mathbf{X}$  is taken as a potential adversarial example  $\tilde{\mathbf{X}}$ . Then, question  $\mathbf{Q}$  is systematically altered for  $n$  number of times by in turn deleting the token  $w_i$ . Each altered question is then passed to the victim model to predict the answer span based on the highest probability values of the start and end tokens. Next, the model predictions for the initial question (i.e.,  $\mathbf{Y}$ ) and that of each altered question (i.e.,  $\tilde{\mathbf{Y}}$ ) are compared. Accordingly, the importance score of the start and end tokens of the answer span  $I_{w_i} = (I_{w_i}^s, I_{w_i}^e)$  for each altered question is computed by either line 5 (i.e., in the case of equality) or line 7 (i.e., otherwise).

$P_{\mathbf{Y}}(\mathbf{X})$  and  $P_{\tilde{\mathbf{Y}}}(\mathbf{X})$  respectively represent the probability values of the start and end tokens of the answer span provided by the ground truth label  $\mathbf{Y}$  and that of the label  $\tilde{\mathbf{Y}}$  predicted by the attacked model for the original question  $\mathbf{X}$ . Similarly,  $P_{\mathbf{Y}}(\mathbf{X}_{\setminus w_i})$  and  $P_{\tilde{\mathbf{Y}}}(\mathbf{X}_{\setminus w_i})$  respectively represent the probability values of the start and end tokens of the answers predicted by the original and attacked model for the perturbed question, in which  $w_i$  has been omitted from the original question.

In line 10, a set of  $\mathbf{W}$  of all words  $w_i \in \mathbf{Q}$  is created and sorted by the descending order of their importance score (i.e., using  $I_{w_i}^s$  or  $(I_{w_i}^s + I_{w_i}^e)/2$ ). In our experiments, we have chosen  $I_{w_i}^s$  to compute the importance score.

**Word Transformation (line 12-34)** In lines 12-14, using the Cosine similarity metric, a set of candidates CANDIDATES is created by extracting the top  $N$  synonyms of word  $w_j$  with the same part of speech as that of  $w_j$ .

In lines 15 to 23, each word  $w_j$  is in turn substituted by a candidate (i.e.,  $c_k$ ) to create an altered example (i.e.,  $\mathbf{X}'$ ). Among all the candidates, those that cause the similarity between the potential adversarial question (i.e.,  $\tilde{\mathbf{Q}}$ ) and the altered question (i.e.,  $\mathbf{Q}'$ ) to exceed a predefined threshold (that we empirically set it to 0.7), are considered as final candidates. Each final candidate along with its predicted label (i.e.,  $\mathbf{Y}_k$ ) and the probability values of its start and end tokens (i.e.,  $\mathbf{P}_k$ ) are stored.

In lines 24-33, at first, the eligibility of each final candidate as an adversarial example is determined by computing an  $\alpha$  value for the candidate and comparing the value against a predefined threshold  $\delta$ . If a candidate modifies the initial question  $\mathbf{Q}$  in a way that results in an altered question  $\mathbf{Q}'$  having the maximum semantic similarity with  $\mathbf{Q}$ , then  $\mathbf{Q}'$  will be chosen as an adversarial question. However,

if a candidate does not satisfy this condition, one of the final candidates with the least confidence score is instead selected.

In RobustQA, we modified TextFooler (Jin et al., 2020), VIPER (Eger et al., 2019; He et al., 2021), Genetic (Alzantot et al., 2018), BERT Attack (Li et al., 2020), PWWS (Ren et al., 2019), SememePSO (Zang et al., 2020), TextBugger (Li et al., 2018), SCPN (Iyyer et al., 2018), and DeepWordBug (Gao et al., 2018) algorithms to be compatible with the QA systems. Note that all the mentioned modifications preserve the nature of these attack algorithms.

## 4 The RobustQA Framework

We have developed a new attack framework named RobustQA for applying text adversarial attack algorithms to QA systems. This framework is an extension of OpenAttack (Zeng et al., 2021), which has been designed for implementing text classification adversarial attacks. RobustQA consists of seven modules, depicted in Figure A.1:

**Tokenizer.** The tokenizer module of RobustQA supports multiple tokenization approaches, including word-, sub-word-, and character-level tokenization. It maintains the consistency between the tokenization of the original sample and that of the adversarial one, enabling the effective evaluation of the attack algorithms. Furthermore, it currently supports the Stanford question answering dataset (SQuAD) dataset (Rajpurkar et al., 2016). However, it can be extended to support any other QA datasets, such as TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2017), etc.

**Victim Model.** The victim model module supports the QA-based models. An extended version of this module is implemented to integrate HuggingFace Transformer-based models\*. This module contains multiple methods required for executing different adversarial attack scenarios in RobustQA. These methods can be overridden or extended for any desired customized attack, as they have access to all the sub-layers of the model’s output and perform their operation as middleware.

**Goals.** The primary target of the goal module is to determine if an input sample is eligible as an adversarial candidate. The candidate sample is regarded as an eligible one if it can confuse the victim model and diminish its performance in terms of EM or F1 score metrics. Defining a custom goal

\*<https://huggingface.co/>

for new QA attacks is achievable by extending the goal module.

**Metrics.** The evaluation metrics of the attack scenarios can be selected or extended with this module. As discussed in Appendix B, the evaluation metrics specific to the QA task (i.e., EM and F1 score) are enabled by default. Other metrics such as edit distance, fluency, grammatical errors, modification rate, and semantic similarity are available for selection.

**Attacker.** The attacker is an abstract module with a default implementation of all the required tools and logic to define an attack algorithm on a given QA victim model. Based on the F1 score metric and a predefined threshold value, an attack goal specific to the given QA task is defined and used as a criterion to determine the adversary potential of different input examples. The primary method of creating a custom QA attack algorithm is to extend the QA attacker module. Various types of adversarial attack algorithms are derived from this module in RobustQA, ready for experimentation.

**Attack Selector.** The attack selector module facilitates the initiation of an attack scenario. This module enables effortless selection and instantiation of the victim model, tokenizer, dataset, attacker, and evaluation metrics. It also performs data sampling and preparation. An attack scenario is easily configured by passing the preferred settings to the attack selector module. Further comprehensive analysis of the attack algorithms is possible by providing additional customized metrics to this module.

**Evaluator.** The execution and evaluation of the QA attack algorithms take place in the evaluator module. Attacks performance is evaluated from different aspects: (1) the attack success rate indicates the percentage of the attacks that fool the victim model and produce false predictions; (2) the modification rate is the percentage of the modified tokens in an adversarial example compared to the input example; (3) the fluency of adversarial examples are computed by perplexity by GPT-2 (Radford et al., 2019); (4) the grammatical errors of each adversarial example is compared to that of the original example, using an available language tool; (5) the semantic similarity between an input example and an adversarial example is computed using a universal sentence encoder (Cer et al., 2018); and finally (6) the average time devoted to the query and attack execution is used to measure the efficacy

of different attacks.

## 5 Toolkit Usage

The RobustQA interface empowers users to execute attack scenarios either programmatically, utilizing the Python programming language, or via a command-line prompt. Appendix D demonstrates an example of the toolkit usage through command-line interface and code. Moreover, some adversarial examples generated by different attack algorithms are depicted in Appendix E.

## 6 Experiments

Utilizing RobustQA, we have evaluated the performance of six different adversarial attack algorithms on the large uncased Bidirectional Encoder Representations from Transformer (BERT) model (Devlin et al., 2019) using the SQuAD dataset, explained in Appendix C. Moreover, we have augmented the training set of SQuAD with an additional 10% adversarial examples generated through the BertAttack algorithm to evaluate the robustness of a given victim model trained by the adversarial training technique. In these experiments, we have considered multiple metrics to evaluate the quality of generated adversarial examples. The results of our experiments with RobustQA and our system setup are presented in Appendices F and G, respectively.

## 7 Conclusion

In this article, we showed the effect of various textual adversarial attack algorithms in character, word, and sentence levels on QA systems. We also developed an open-source framework, named RobustQA, for the field of textual adversarial attack on QA systems, which consists of seven primary modules. This new framework offers different features that are easily customizable for applying existing or designing new algorithms, along with efficient analysis of attack scenarios. As our future work, this framework can be further extended to include other attack algorithms. We can also provide more functions and tools for further research in the context of attacks and defense within QA systems. The source code and documentation of RobustQA are available at <https://github.com/mirbostani/RobustQA>.

## Limitations

Although RobustQA is reliable for implementing and evaluating textual adversarial attacks on QA models, a limitation may arise in certain attack algorithms due to their high resource requirements. Specifically, in some cases, the execution of the attack algorithms requires a high level of GPU resources and CPU iterations. Like many other deep learning algorithms, adversarial text generation and adversarial training heavily rely on GPU resources. As the augmented training set grows, the mentioned procedures demand a substantial share of GPU power. This requirement imposed some constraint on the extent of our experiments.

Due to the intricacies of the QA domain and the diverse nature of attacks in this domain, it was not feasible for us to seamlessly integrate all of them. Some algorithms could perfectly align with specific QA architectures, while others might require some customizations. Although the required tools for implementing any adversarial attack algorithm can be embedded within the RobustQA framework, the challenge of adapting all the attack algorithms hindered the variety of our experiments conducted in this study.

## Ethics Statement

The primary focus of this study has been on enhancing the robustness of NLP models to make these models less vulnerable to potential misuse. We foresee no ethical issues arising from the algorithms and techniques introduced in this study. All the datasets, tools, and libraries employed in this study are open-source and publicly accessible.

## References

- Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. [Generating natural language adversarial examples](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.
- Yasaman Boreshban, Seyed Morteza Mirbostani, Gholamreza Ghassem-Sani, Seyed Abolghasem Mirroshandel, and Shahin Amiriparian. 2023. Improving question answering performance using knowledge distillation and active learning. *Engineering Applications of Artificial Intelligence*, 123:106137.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. [Universal sentence encoder for English](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, 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.
- Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. [HotFlip: White-box adversarial examples for text classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.
- Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. [Text processing like humans do: Visually attacking and shielding NLP systems](#). 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 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.
- Wee Chung Gan and Hwee Tou Ng. 2019. [Improving the robustness of question answering systems to question paraphrasing](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.
- Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.
- Siddhant Garg and Goutham Ramakrishnan. 2020. [BAE: BERT-based adversarial examples for text classification](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.
- Yotam Gil, Yoav Chai, Or Gorodissky, and Jonathan Berant. 2019. [White-to-black: Efficient distillation of black-box adversarial attacks](#). 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 1373–1379, Minneapolis, Minnesota. Association for Computational Linguistics.

- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. 2020. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv preprint arXiv:2001.05574*.
- Xuanli He, Lingjuan Lyu, Qiongfai Xu, and Lichao Sun. 2021. Model extraction and adversarial transferability, your bert is vulnerable! *arXiv preprint arXiv:2103.10013*.
- Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. *arXiv preprint arXiv:2101.10579*.
- Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. [Adversarial example generation with syntactically controlled paraphrase networks](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.
- 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.
- Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.
- 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.
- Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. [The NarrativeQA reading comprehension challenge](#). *Transactions of the Association for Computational Linguistics*, 6:317–328.
- Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.
- Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. [BERT-ATTACK: Adversarial attack against BERT using BERT](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773.
- John Morris, Eli Liland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. [TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.
- Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Amrith Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. 2018. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*.
- Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Chang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. 2016. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*.
- Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. [Combating adversarial misspellings with robust word recognition](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. 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.
- 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.
- Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*.
- Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. [Generating natural language adversarial examples through probability weighted word saliency](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. [Semantically equivalent adversarial rules for debugging NLP models](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Walter Simoncini and Gerasimos Spanakis. 2021. [SeqAttack: On adversarial attacks for named entity recognition](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 308–318, Online and Punta Cana, Dominican Republic. 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.

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.

Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li. 2020. [T3: Tree-autoencoder constrained adversarial text generation for targeted attack](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6134–6150, Online. Association for Computational Linguistics.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178.

Ziqing Yang, Yiming Cui, Chenglei Si, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. 2021. [Adversarial training for machine reading comprehension with virtual embeddings](#). In *Proceedings of \*SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 308–313, Online. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. [Word-level textual adversarial attacking as combinatorial optimization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan

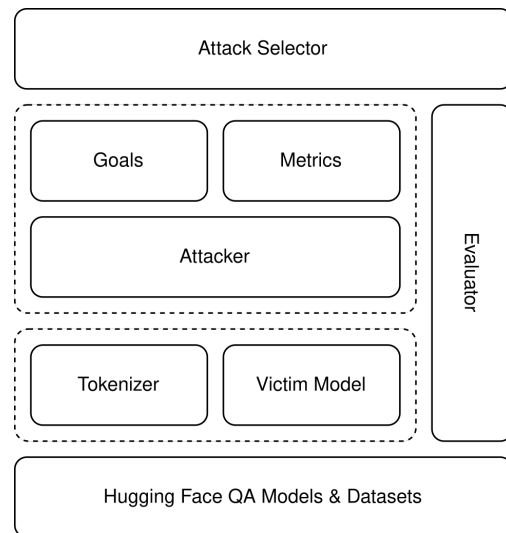


Figure A.1: The architecture of RobustQA.

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, Online. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.

## A Architecture

RobustQA is the first open-source framework for textual adversarial attack analysis in QA systems. As shown in Figure A.1, it consists of seven modules: Tokenizer, Victim Model, Goals, Metrics, Attacker, Attack Selector, and Evaluator. Currently, six different adversarial attack algorithms have been implemented in this framework.

## B Evaluation Metrics

For the evaluation purpose, we have employed EM and F1 score criteria, which are regarded as the standard metrics for evaluating QA systems (Rajpurkar et al., 2016). The F1 measure represents the average overlap between the ground truth and the predicted answers. On the other hand, the EM measure demonstrates the percentage of those responses that exactly match the ground truth answer.

---

```

MODEL="bert-large-uncased-whole-word-masking\
-finetuned-squad"
python qa.py \
  --use_cuda \
  --victim_model_or_path "$MODEL" \
  --victim_tokenizer_or_path "$MODEL" \
  --dataset "squad" \
  --dataset_split "validation[0:1000]" \
  --attack_recipe "textfooler" \
  --batch_size 8 \
  --language "english" \
  --use_metric_f1_score \
  --use_metric_exact_match \
  --use_metric_edit_distance \
  --use_metric_fluency \
  --use_metric_grammatical_errors \
  --use_metric_modification_rate \
  --use_metric_semantic_similarity \
  --use_metric_jaccard_char_similarity \
  --use_metric_jaccard_word_similarity

```

---

Figure D.1: Executing the TextFooler attack on BERT<sub>LARGE</sub> via command-line interface.

## C Datasets

The SQuAD v1.1, introduced in 2016 by (Rajpurkar et al., 2016), is a reading comprehension dataset containing 107,785 question-answer pairs derived from 536 Wikipedia documents. In this version of SQuAD, the answer to each question is a span of the text extracted from the associated paragraph in the document. The training and validation datasets contain 87,599 and 10,570 question-answer pairs, respectively.

## D Usage Examples

Figure D.1 is an example demonstrating the usage of the RobustQA framework through the command-line interface. The TextFooler attack algorithm is executed on the BERT<sub>LARGE</sub> model employing the first 1000 validation examples of the SQuAD dataset.

The results of the TextFooler attack on the BERT<sub>LARGE</sub> model along with all the computed metric values are summarized in Figure D.2.

The same attack scenario can be executed by code using the Python programming language demonstrated in Figure D.3.

## E Adversarial Examples

In this section, the generated adversarial examples of three attack algorithms are presented. The original and adversary questions are depicted in Table E.1. Other fields of the generated adversarial

| Summary                         |         |
|---------------------------------|---------|
| Total Attacked Instances:       | 1000    |
| Successful Instances:           | 230     |
| Attack Success Rate:            | 0.23    |
| Avg. Running Time:              | 0.30072 |
| Total Query Exceeded:           | 0       |
| Avg. Victim Model Queries:      | 85.819  |
| Avg. Exact Match (Orig):        | 57.1    |
| Avg. Exact Match (Adv):         | 46.2    |
| Avg. F1 Score (Orig):           | 72.305  |
| Avg. F1 Score (Adv):            | 59.183  |
| Avg. Levenshtein Edit Distance: | 1.6609  |
| Avg. Fluency (ppl):             | 1765.1  |
| Avg. Grammatical Errors:        | 0       |
| Avg. Word Modif. Rate:          | 0.15899 |
| Avg. Semantic Similarity:       | 0.84477 |
| Avg. Jaccard Char Similarity:   | 0.91577 |
| Avg. Jaccard Word Similarity:   | 0.46207 |

Figure D.2: The results of the TextFooler attack on BERT<sub>LARGE</sub>.

examples, such as "context" and "answers", are the same as the original instance from the SQuAD dataset.

## F Results

In this section, we present the evaluation results of six different adversarial attack algorithms implemented with the RobustQA framework. The experiments have been performed on the BERT<sub>LARGE</sub> victim model employing the first 1000 validation examples from SQuAD dataset. The original F1 score and EM measure of the victim model, calculated before carrying out the attack algorithms, are 72.3% and 57.1%, respectively. The victim model's performance results are summarized in Table F.1.

Furthermore, we have expanded the training dataset of SQuAD by combining an additional 10% of adversarial examples generated through the BertAttack algorithm using the RobustQA framework. The augmented training dataset is used in training the BERT<sub>LARGE</sub> model. Finally, to demonstrate the effect of adversarial training on QA models, we have evaluated this model on six different adversarial attack algorithms using RobustQA. The results of the evaluation on 1000 validation examples of from the SQuAD dataset is shown in Table F.2.

## G Experiment Setup

The computational experiments in this study were conducted on a system with an Intel Core i7-8700K CPU 3.70GHz 6-Core, a GeForce GTX 1080 8GB vRAM, and 64GB of RAM.

---

```

from qa.victim.question_answering.transformers import TransformersQuestionAnswering
from qa.attackers.textfooler import TextFoolerAttacker
from qa.metric import QAScore, EditDistance
from qa.attack_eval import AttackEval
from transformers import AutoTokenizer, AutoModelForQuestionAnswering
from datasets import load_dataset

model_name = "bert-large-uncased-whole-word-masking-finetuned-squad"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
victim = TransformersQuestionAnswering(
    model=model,
    tokenizer=tokenizer,
    embedding_layer=model.bert.embeddings.word_embeddings,
    device="cuda",
    max_length=512,
    truncation=True,
    padding=True,
    batch_size=8,
    lang="english"
)
dataset = load_dataset("squad", split="validation[0:1000]")
attacker = TextFoolerAttacker(tokenizer=victim.tokenizer, max_length=512)
metrics = [
    QAScore(victim, "f1", "orig"),
    QAScore(victim, "f1", "adv"),
    QAScore(victim, "em", "orig"),
    QAScore(victim, "em", "adv"),
    EditDistance()
]
evaluator = AttackEval(attacker, victim, metrics=metrics)
evaluator.eval(dataset, visualize=True, progress_bar=True)

```

---

Figure D.3: Executing the TextFooler attack on BERT<sub>LARGE</sub> via Python code.

|                   | Question  |
|-------------------|---|
| <b>Original</b>   | What part of Luther's career was one of his most productive?                        |
| <b>TextFooler</b> | what <b>portion</b> of luther's <b>calling</b> was one of his most productive?      |
| <b>Original</b>   | What high maintenance part did Tesla's AC motor not require?                        |
| <b>Sememe PSO</b> | what <b>in</b> maintenance <b>percentage</b> did tesla's ac motor not <b>call</b> ? |
| <b>Original</b>   | Who was the main performer at this year's halftime show?                            |
| <b>PWWS</b>       | who was the <b>principal</b> performer at this year's halftime show?                |

Table E.1: Adversarial examples of three attack algorithms are showcased. The TextFooler, Sememe PSO, and PWWS algorithms are executed on the original question of the given instances from the validation set of the SQuAD dataset. The modified segments influenced by each attack are highlighted using bold text.



|                           | <b>TextFooler</b> | <b>PWWS</b> | <b>Genetic</b> | <b>Sememe PSO</b> | <b>BertAttack</b> | <b>DeepWordBug</b> |
|---------------------------|-------------------|-------------|----------------|-------------------|-------------------|--------------------|
| Successful Instances      | 230               | 377         | 710            | 639               | 794               | 484                |
| Attack Success Rate       | 0.23              | 0.37        | 0.71           | 0.63              | 0.79              | 0.48               |
| Running Time              | 0.30              | 0.04        | 6.78           | 0.38              | 0.38              | 0.02               |
| Victim Model Queries      | 85.81             | 61.33       | 1714.90        | 106.26            | 91.18             | 15.16              |
| Exact Match (Original)    | 57.1              | 57.1        | 57.1           | 57.1              | 57.1              | 57.1               |
| Exact Match (Adversary)   | 46.2              | 39.4        | 17.7           | 23.5              | 12.5              | 31.6               |
| F1 Score (Original)       | 72.3              | 72.3        | 72.3           | 72.3              | 72.3              | 72.3               |
| F1 Score (Adversary)      | 59.1              | 50.8        | 23.2           | 30.3              | 17.2              | 40.7               |
| Levenshtein Edit Distance | 1.66              | 1.93        | 2.37           | 1.98              | 2.19              | 4.09               |
| Fluency                   | 1765.1            | 2101.3      | 2250.3         | 1719.6            | 941.7             | 1274.6             |
| Word Modification Rate    | 0.15              | 0.16        | 0.22           | 0.19              | 0.32              | 0.37               |
| Semantic Similarity       | 0.84              | 0.79        | 0.76           | 0.79              | 0.83              | 0.55               |
| Jaccard Char Similarity   | 0.91              | 0.90        | 0.89           | 0.89              | 0.89              | 0.80               |
| Jaccard Word Similarity   | 0.46              | 0.43        | 0.41           | 0.43              | 0.44              | 0.34               |

Table F.1: QA adversarial attacks evaluation on BERT<sub>LARGE</sub> using RobustQA.

|                           | <b>TextFooler</b> | <b>PWWS</b> | <b>Genetic</b> | <b>Sememe PSO</b> | <b>BertAttack</b> | <b>DeepWordBug</b> |
|---------------------------|-------------------|-------------|----------------|-------------------|-------------------|--------------------|
| Successful Instances      | 263               | 394         | 667            | 620               | 746               | 566                |
| Attack Success Rate       | 0.26              | 0.39        | 0.66           | 0.62              | 0.74              | 0.56               |
| Running Time              | 0.29              | 0.04        | 7.51           | 0.37              | 0.39              | 0.02               |
| Victim Model Queries      | 83.56             | 61.19       | 1845.8         | 104.49            | 94.56             | 15.24              |
| Exact Match (Original)    | 47.8              | 47.8        | 47.8           | 47.8              | 47.8              | 47.8               |
| Exact Match (Adversary)   | 38.7              | 34.8        | 17.0           | 21.9              | 13.5              | 24.0               |
| F1 Score (Original)       | 64.2              | 64.2        | 64.2           | 64.2              | 64.2              | 64.2               |
| F1 Score (Adversary)      | 52.0              | 47.1        | 23.7           | 29.9              | 19.7              | 31.9               |
| Levenshtein Edit Distance | 1.54              | 1.85        | 2.37           | 1.90              | 2.08              | 4.62               |
| Fluency                   | 1130.7            | 1879.5      | 1642.2         | 1445.1            | 1001.3            | 1174.8             |
| Word Modification Rate    | 0.15              | 0.15        | 0.23           | 0.19              | 0.31              | 0.42               |
| Semantic Similarity       | 0.86              | 0.80        | 0.77           | 0.80              | 0.83              | 0.54               |
| Jaccard Char Similarity   | 0.92              | 0.91        | 0.89           | 0.90              | 0.90              | 0.79               |
| Jaccard Word Similarity   | 0.46              | 0.45        | 0.42           | 0.44              | 0.45              | 0.31               |

Table F.2: QA adversarial training evaluation on BERT<sub>LARGE</sub> trained on SQuAD and 10% of adversarial examples generated by the BertAttack algorithm in RobustQA.

# Kandinsky: an Improved Text-to-Image Synthesis with Image Prior and Latent Diffusion

Anton Razzhigaev<sup>1,2</sup>, Arseniy Shakhmatov<sup>1</sup>, Anastasia Maltseva<sup>1</sup>, Vladimir Arkhipkin<sup>1</sup>, Igor Pavlov<sup>1</sup>, Ilya Ryabov<sup>1</sup>, Angelina Kuts<sup>1</sup>, Alexander Panchenko<sup>1</sup>, Andrey Kuznetsov<sup>1</sup>, and Denis Dimitrov<sup>1</sup>

<sup>1</sup>AIRI, <sup>2</sup>Center for Artificial Intelligence Technology  
{razzhigaev, kuznetsov, dimitrov}@airi.net

## Abstract

Text-to-image generation is a significant domain in modern computer vision and has achieved substantial improvements through the evolution of generative architectures. Among these, there are diffusion-based models that have demonstrated essential quality enhancements. These models are generally split into two categories: pixel-level and latent-level approaches. We present Kandinsky<sup>1</sup>, a novel exploration of latent diffusion architecture, combining the principles of the image prior models with latent diffusion techniques. The image prior model is trained separately to map text embeddings to image embeddings of CLIP. Another distinct feature of the proposed model is the modified MoVQ implementation, which serves as the image autoencoder component. Overall, the designed model contains 3.3B parameters. We also deployed a user-friendly demo system that supports diverse generative modes such as text-to-image generation, image fusion, text and image fusion, image variations generation, and text-guided inpainting/outpainting. Additionally, we released the source code and checkpoints for the Kandinsky models. Experimental evaluations demonstrate a FID score of 8.03 on the COCO-30K dataset, marking our model as the top open-source performer in terms of measurable image generation quality.

## 1 Introduction

In quite a short period of time, generative abilities of text-to-image models have improved substantially, providing users with photorealistic quality, near real-time inference speed, a great number of applications and features, including simple easy-to-use web-based platforms and sophisticated AI graphics editors.

This paper presents our unique investigation of latent diffusion architecture design, offering a fresh

<sup>1</sup>The system is named after Wassily Kandinsky, a famous painter and an art theorist.

and innovative perspective on this dynamic field of study. First, we describe the new architecture of Kandinsky and its details. The demo system with implemented features of the model is also described. Second, we show the experiments, carried out in terms of image generation quality and come up with the highest FID score among existing open-source models. Additionally, we present the rigorous ablation study of prior setups that we conducted, enabling us to carefully analyze and evaluate various configurations to arrive at the most effective and refined model design.

Our **contributions** are as follows:

- We present the first text-to-image architecture designed using a combination of image prior and latent diffusion.
- We demonstrate experimental results comparable to the state-of-the-art (SotA) models such as Stable Diffusion, IF, and DALL-E 2, in terms of FID metric and achieve the SotA score among all existing open source models.
- We provide a software implementation of the proposed state-of-the-art method for text-to-image generation, and release pre-trained models, which is unique among the top-performing methods. Apache 2.0 license makes it possible to use the model for both non-commercial and commercial purposes.<sup>2 3</sup>
- We create a web image editor application that can be used for interactive generation of images by text prompts (English and Russian languages are supported) on the basis of the proposed method, and provides inpainting/outpainting functionality.<sup>4</sup> The video demonstration is available on YouTube.<sup>5</sup>

<sup>2</sup><https://github.com/ai-forever/Kandinsky-2>

<sup>3</sup><https://huggingface.co/kandinsky-community>

<sup>4</sup><https://fusionbrain.ai/en/editor>

<sup>5</sup><https://www.youtube.com/watch?v=c7zHPc59cWU>

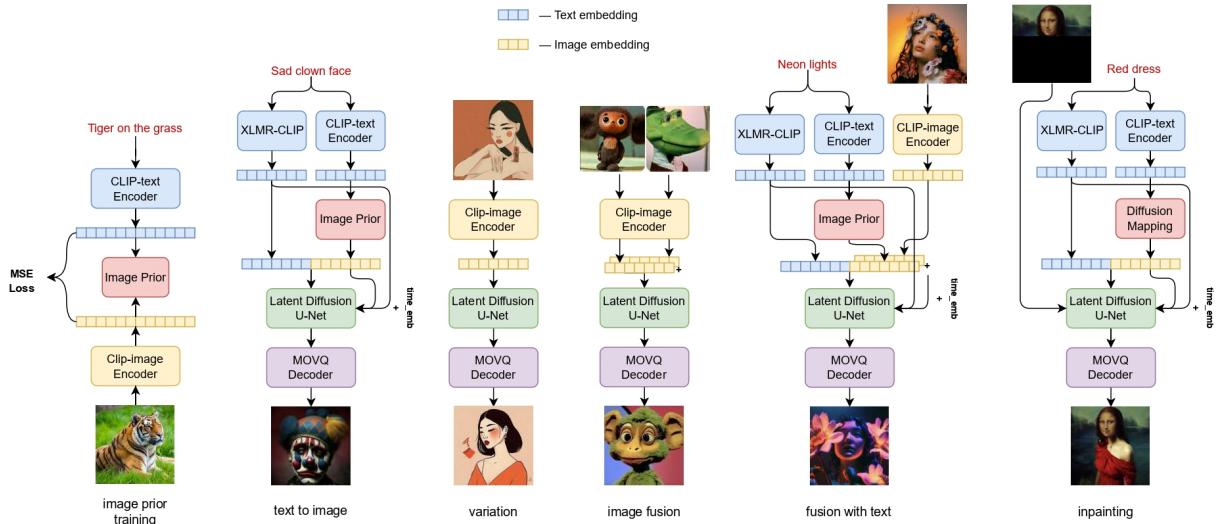


Figure 1: Image prior scheme and inference regimes of the Kandinsky model.

## 2 Related Work

Early text-to-image generative models, such as DALL-E (Ramesh et al., 2021) and CogView (Ding et al., 2021), or later Parti (Yu et al., 2022) employed autoregressive approaches but often suffered from significant content-level artifacts. This led to the development of a new breed of models that utilized the diffusion process to enhance image quality. Diffusion-based models, such as DALL-E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022b), and Stable Diffusion<sup>6</sup>, have since become cornerstones in this domain. These models are typically divided into pixel-level (Ramesh et al., 2022; Saharia et al., 2022b) and latent-level (Rombach et al., 2022) approaches.

This surge of interest has led to the design of innovative approaches and architectures, paving the way for numerous applications based on open-source generative models, such as DreamBooth (Ruiz et al., 2023) and DreamPose (Karras et al., 2023). These applications exploit image generation techniques to offer remarkable features, further fueling the popularity and the rapid development of diffusion-based image generation approaches.

This enabled a wide array of applications like 3D object synthesis (Poole et al., 2023; Tang et al., 2023; Lin et al., 2022; Chen et al., 2023), video generation (Ho et al., 2022b; Luo et al., 2023; Ho et al., 2022a; Singer et al., 2023; Blattmann et al., 2023; Esser et al., 2023), controllable image editing (Hertz et al., 2023; Parmar et al., 2023; Liew et al., 2022; Mou et al., 2023; Lu et al., 2023), and more,

<sup>6</sup><https://github.com/CompVis/stable-diffusion>

which are now at the forefront of this domain.

Diffusion models achieve state-of-the-art results in image generation task both unconditional (Ho et al., 2020; Nichol and Dhariwal, 2021) and conditional (Peebles and Xie, 2022). They beat GANs (Goodfellow et al., 2014) by generating images with better scores of fidelity and diversity without adversarial training (Dhariwal and Nichol, 2021). Diffusion models also show the best performance in various image processing tasks like inpainting, outpainting, and super-resolution (Batzolis et al., 2021; Saharia et al., 2022a).

Text-to-image diffusion models have become a popular research direction due to the high performance of diffusion models and the ability to simply integrate text conditions with the classifier-free guidance algorithm (Ho and Salimans, 2022). Early models like GLIDE (Nichol et al., 2022), Imagen (Saharia et al., 2022b), DALL-E 2 (Ramesh et al., 2022) and eDiff-I (Balaji et al., 2022) generate low-resolution image in pixel space and then upsample it with another super-resolution diffusion models. They are also using different text encoders, large language model T5 (Raffel et al., 2020) in Imagen, CLIP (Radford et al., 2021) in GLIDE and DALL-E 2.

## 3 Demo System

We implemented a set of user-oriented solutions where Kandinsky model is embedded as a core imaging service. It has been done due to a variety of inference regimes, some of which need specific front-end features to perform properly. Overall, we implemented two main inference resources: Tele-

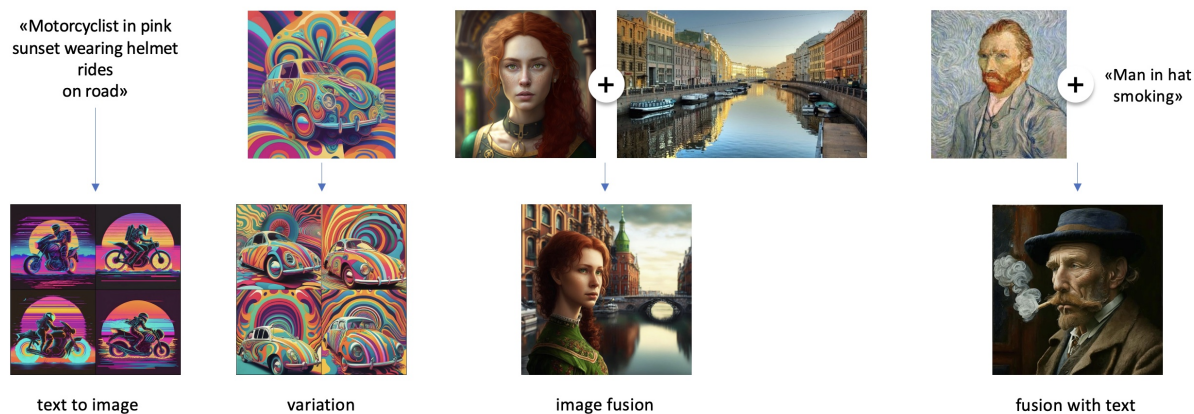


Figure 2: Examples of inference regimes using Kandinsky model.

gram bot and FusionBrain website.

FusionBrain represents a web-based image editor with such features as loading and saving images, sliding location window, erasing tools, zooming in/out, various styles selector, etc. (cf. Figure 3). In terms of image generation, the three following options are implemented on this side:

- text-to-image generation – user inputs a text prompt in Russian or English, then selects an aspect-ratio from the list (9:16, 2:3, 1:1, 16:9, 3:2), and the system generates an image;
- inpainting – using the specific erasing tool, user can remove any arbitrary input image part and fill it, guided by a text prompt or without any guidance;
- outpainting – input image can be extended with a sliding window that can be used as a mask for the following generation (if the window intersects any imaged area, then the empty window part is generated with or without text prompt guidance).

Inpainting and outpainting options are the main image editing features of the model. Architectural details about these generation types can also be found in Figure 1.

Telegram bot contains the following image generation features (cf. Figure 2):

- text-to-image generation;
- image and text fusion – user inputs an image and a text prompt to create a new image guided by this prompt;
- image fusion – user inputs an image as the main one and another 'guiding' image, and the system generates their fusion;

- image variations – user inputs an image, and the system generates several new images similar to the input one.

## 4 Kandinsky Architecture

In our work, we opted to deliver state-of-the-art text-to-image synthesis. In the initial stages of our research, we experimented with multilingual text encoders, such as mT5 (Xue et al., 2021), XLMR (Conneau et al., 2020), XLMR-CLIP<sup>7</sup>, to facilitate robust multilingual text-to-image generation. However, we discovered that using the CLIP-image embeddings instead of standalone text encoders resulted in improved image quality. As a result, we adopted an image prior approach, utilizing diffusion and linear mappings between text and image embedding spaces of CLIP, while keeping additional conditioning with XLMR text embeddings. That is why Kandinsky uses two text encoders: CLIP-text with image prior mapping and XLMR. We have set these encoders to be frozen during the training phase.

The significant factor that influenced our design choice was the efficiency of training latent diffusion models, as compared to pixel-level diffusion models (Rombach et al., 2022). This led us to focus our efforts on the latent diffusion architecture. Our model essentially comprises three stages: text encoding, embedding mapping (image prior), and latent diffusion.

The construction of our model involves three primary steps: text encoding, embedding mapping (image prior), and latent diffusion. At the embedding mapping step, which we also refer to as the

<sup>7</sup><https://github.com/FreddeFrallan/Multilingual-CLIP>

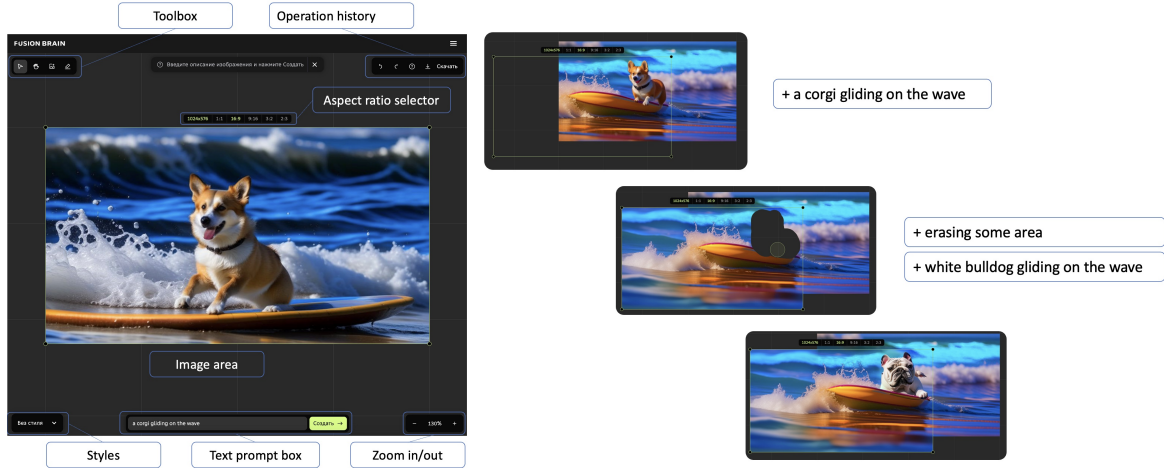


Figure 3: Kandinsky web interface for “a corgi gliding on the wave”: generation (left) and in/outpainting (right).

Table 1: Proposed architecture comparison by FID on COCO-30K validation set on  $256 \times 256$  resolution. \* For the IF model we reported reproduced results on COCO-30K, but authors provide FID of 7.19.

| Model                                    | FID-30K     |
|--|-------------|
| <i>Open Sourced Technologies</i>         |             |
| <b>Kandinsky (Ours)</b>                  | <b>8.03</b> |
| Stable Diffusion 2.1 (2022) <sup>8</sup> | 8.59        |
| GLIDE <sup>8</sup> (Nichol et al., 2022) | 12.24       |
| IF* (2023) <sup>12</sup>                 | 15.10       |
| Kandinsky 1.0 (2022) <sup>9</sup>        | 15.40       |
| ruDALL-E Malevich (2022) <sup>9</sup>    | 20.00       |
| GLIGEN <sup>10</sup> (Li et al., 2023)   | 21.04       |
| <i>Proprietary Technologies</i>          |             |
| <b>eDiff-I (Balaji et al., 2022)</b>     | <b>6.95</b> |
| Imagen (Saharia et al., 2022b)           | 7.27        |
| GigaGAN (Kang et al., 2023)              | 9.09        |
| DALL-E 2 (Ramesh et al., 2022)           | 10.39       |
| DALL-E (Ramesh et al., 2021)             | 17.89       |

image prior, we use the transformer-encoder model. This model was trained from scratch with a diffusion process on text and image embeddings provided by the CLIP-ViT-L14 model. A noteworthy feature in our training process is the use of element-wise normalization of visual embeddings. This normalization is based on full-dataset statistics and leads to faster convergence of the diffusion process. We implemented inverse normalization to revert to the original CLIP-image embedding space in the inference stage.

The image prior model is trained on text and image embeddings, provided by the CLIP models.

<sup>8</sup><https://github.com/Stability-AI/stablediffusion>

<sup>9</sup><https://github.com/ai-forever/ru-dalle>

<sup>10</sup><https://github.com/gligen/GLIGEN>

We conducted a series of experiments and ablation studies on the specific architecture design of the image prior model (Table 3, Figure 6). The model with the best human evaluation score is based on a 1D-diffusion and standard transformer-encoder with the following parameters: num\_layers=20, num\_heads=32, and hidden\_size=2048.

The latent diffusion part employs a UNet model along with a custom pre-trained autoencoder. Our diffusion model uses a combination of multiple condition signals: CLIP-image embeddings, CLIP-text embeddings, and XLMR-CLIP text embeddings. CLIP-image and XLMR-CLIP embeddings are merged and utilized as an input to the latent diffusion process. Also, we conditioned the diffusion process on these embeddings by adding all of them to the time-embedding. Notably, we did not skip the quantization step of the autoencoder during diffusion inference as it leads to an increase in the diversity and the quality of generated images (cf. Figure 4). In total, our model comprises 3.3 B parameters (Table 2).

Table 2: Kandinsky model parameters.

| Architecture part            | Params | Freeze |
|------------------------------|--------|--------|
| Diffusion Mapping            | 1B     | False  |
| CLIP image encoder (ViT-L14) | 427M   | True   |
| CLIP text encoder            | 340M   | True   |
| Text encoder (XLM-R-L)       | 560M   | True   |
| Latent Diffusion UNet        | 1.22B  | False  |
| MoVQ image autoencoder       | 67M    | True   |

We observed that the image decoding was our main bottleneck in terms of generated image quality; hence, we developed a Sber-MoVQGAN, our custom implementation of MoVQGAN (Zheng

Table 3: Ablation study: FID on COCO-30K validation set on  $256 \times 256$  resolution.

| Setup                             | FID-30K     | CLIP         |
|-----------------------------------|-------------|--------------|
| Diffusion prior with quantization | 9.86        | <b>0.287</b> |
| Diffusion prior w/o quantization  | 9.87        | 0.286        |
| Linear prior                      | <b>8.03</b> | 0.261        |
| Residual prior                    | 8.61        | 0.249        |
| No prior                          | 25.92       | 0.256        |

et al., 2022) with minor modifications. We trained this autoencoder on the LAION HighRes dataset (Schuhmann et al., 2022), obtaining the SotA results in image reconstruction. We released the weights and code for these models under an open source licence<sup>11</sup>. The comparison of our autoencoder with competitors can be found in Table 4.

## 5 Experiments

We sought to evaluate and refine the performance of our proposed latent diffusion architecture in our experimental analysis. To this end, we employed automatic metrics, specifically FID-CLIP curves on the COCO-30K dataset, to obtain the optimal guidance-scale value and compare Kandinsky with competitors (cf. Figure 4). Furthermore, we conducted investigations with various image prior setups, exploring the impact of different configurations on the performance. These setups included: no prior, utilizing text embeddings directly; linear prior, implementing one linear layer; ResNet prior, consisting of 18 residual MLP blocks; and transformer diffusion prior.

An essential aspect of our experiments was the exploration of the effect of latent quantization within the MoVQ autoencoder. We examined the outputs with latent quantization, both enabled and disabled, to better comprehend its influence on image generation quality.

To ensure a comprehensive evaluation, we also included an assessment of the IF model<sup>12</sup>, which is the closest open-source competitor to our proposed model. For this purpose, we computed FID scores for the IF model<sup>13</sup> (Table 1).

However, we acknowledged the limitations of automatic metrics that become obvious when it comes to capturing user experience nuances. Hence, in addition to the FID-CLIP curves, we conducted a blind human evaluation to obtain insightful feed-

<sup>11</sup><https://github.com/ai-forever/MoVQGAN>

<sup>12</sup><https://github.com/deep-floyd/IF>

<sup>13</sup><https://github.com/mseitzer/pytorch-fid>

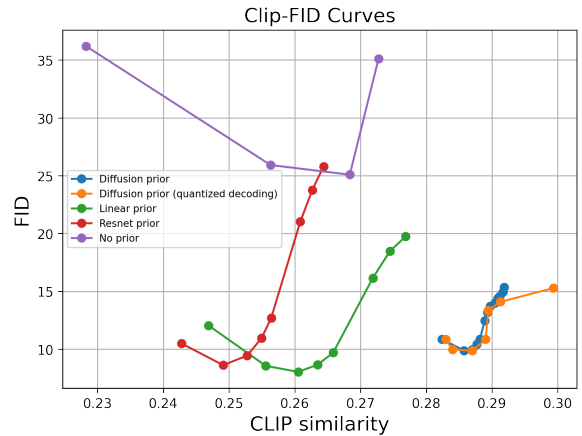


Figure 4: CLIP-FID curves for different setups.



original image prior

cat-500 prior

Figure 5: Image generation results with prompt "astronaut riding a horse" for original image prior and linear prior trained on 500 pairs of images with cats.

back and validate the quality of the generated images from the perspective of human perception based on the DrawBench dataset (Saharia et al., 2022b).

The combination of automatic metrics and human evaluation provides a comprehensive assessment of Kandinsky performance, enabling us to make informed decisions about the effectiveness and usability of our proposed image prior to design.

## 6 Results

Our experiments and evaluations have showcased the capabilities of Kandinsky architecture in text-to-image synthesis. Kandinsky achieved the FID score of 8.03 on the COCO-30K validation set at a resolution of  $256 \times 256$ , which puts it in close competition with the state-of-the-art models, and among the top performers within open-source systems. Our methodical ablation studies further dissected the performance of different configurations: quantization of latent codes in MoVQ slightly improves

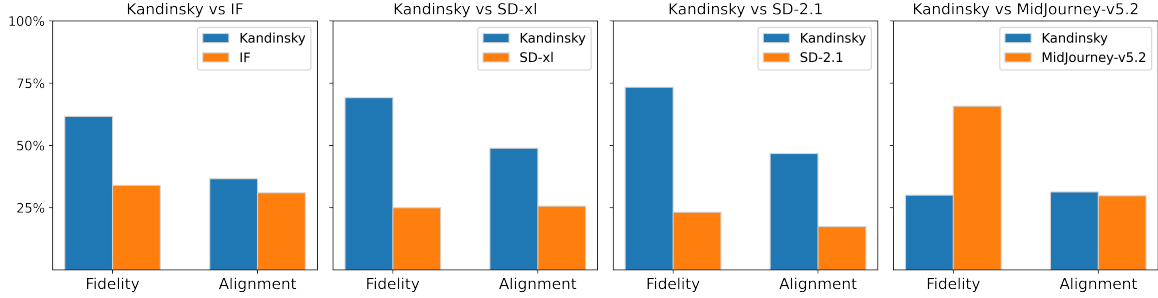


Figure 6: Human evaluation: competitors vs Kandinsky with diffusion prior on Drawbench. The total count of votes is 5000.

Table 4: Sber-MoVQGAN comparison with competitors on ImageNet dataset.

| Model             | Latent size | Num Z | Train steps | FID ↓        | SSIM ↑       | PSNR ↑       | L1 ↓          |
|-------------------|-------------|-------|-------------|--------------|--------------|--------------|---------------|
| ViT-VQGAN*        | 32x32       | 8192  | 500,000     | 1.28         | -            | -            | -             |
| RQ-VAE*           | 8x8x16      | 16384 | 10 epochs   | 1.83         | -            | -            | -             |
| Mo-VQGAN*         | 16x16x4     | 1024  | 40 epochs   | 1.12         | 0.673        | 22.42        | -             |
| VQ CompVis        | 32x32       | 16384 | 971,043     | 1.34         | 0.650        | 23.85        | 0.0533        |
| KL CompVis        | 32x32       | -     | 246,803     | 0.968        | 0.692        | 25.11        | 0.0474        |
| Sber-VQGAN        | 32x32       | 8192  | 1 epoch     | 1.44         | 0.682        | 24.31        | 0.0503        |
| Sber-MoVQGAN 67M  | 32x32       | 1024  | 5,000,000   | 1.34         | 0.704        | 25.68        | 0.0451        |
| Sber-MoVQGAN 67M  | 32x32       | 16384 | 2,000,000   | 0.965        | 0.725        | 26.45        | 0.0415        |
| Sber-MoVQGAN 102M | 32x32       | 16384 | 2,360,000   | 0.776        | 0.737        | 26.89        | 0.0398        |
| Sber-MoVQGAN 270M | 32x32       | 16384 | 1,330,000   | <b>0.686</b> | <b>0.741</b> | <b>27.04</b> | <b>0.0393</b> |

the quality of images (FID 9.86 vs 9.87). The best CLIP score and human-eval score are obtained by diffusion prior.

The best FID score is achieved using Linear Prior. This configuration stands out with the best FID score of 8.03. It is an intriguing outcome: the simplest linear mapping showcased the best FID, suggesting that there might exist a linear relationship between visual and textual embedding vector spaces. To further scrutinize this hypothesis, we trained a linear mapping on a subset of 500 cat images and termed it the "cat prior". Astonishingly, this mapping displayed high proficiency (cf. Figure 5).

## 7 Conclusion

We presented Kandinsky, a system for various image generation and processing tasks based on a novel latent diffusion model. Our model yielded the SotA results among open-sourced systems. Additionally, we provided an extensive ablation study of an image prior to design choices. Our system is equipped with free-to-use interfaces in the form of Web application and Telegram messenger bot. The pre-trained models are available on Hugging Face, and the source code is released under a permissive

license enabling various, including commercial, applications of the developed technology.

In future research, our goal is to investigate the potential of the latest image encoders. We plan to explore the development of more efficient UNet architectures for text-to-image tasks and focus on improving the understanding of textual prompts. Additionally, we aim to experiment with generating images at higher resolutions and to investigate new features extending the model: local image editing by a text prompt, attention reweighting, physics-based generation control, etc. The robustness against generating abusive content remains a crucial concern, warranting the exploration of real-time moderation layers or robust classifiers to mitigate undesirable, e.g. toxic or abusive, outputs.

## 8 Limitations

The current system produces images that appear natural, however, additional research can be conducted to (1) enhance the semantic coherence between the input text and the generated image, and (2) to improve the absolute values of FID and image quality based on human evaluations.

## 9 Ethical Considerations

We performed multiple efforts to ensure that the generated images do not contain harmful, offensive, or abusive content by (1) cleansing the training dataset from samples that were marked to be harmful/offensive/abusive, and (2) detecting abusive textual prompts.

While obvious queries, according to our tests, almost never generate abusive content, technically it is not guaranteed that certain carefully engineered prompts may not yield undesirable content. We, therefore, recommend using an additional layer of classifiers, depending on the application, which would filter out the undesired content and/or use image/representation transformation methods tailored to a given application.

### Acknowledgements

As usual, we would like to thank the anonymous reviewers for their useful comments. We would also like to thank Sergey Markov and his team for helpful feedback and discussions, for collaboration in multimodal dataset collecting, labelling and processing.

### References

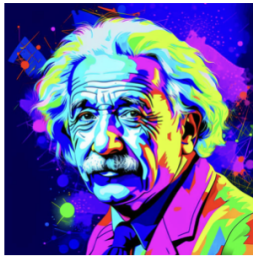
- Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. 2022. [ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers](#).
- Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. 2021. [Conditional image generation with score-based diffusion models](#). *CoRR*, abs/2111.13606.
- Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. 2023. [Align your latents: High-resolution video synthesis with latent diffusion models](#). *CoRR*, abs/2304.08818.
- Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023. [Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation](#). *CoRR*, abs/2303.13873.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.
- Prafulla Dhariwal and Alexander Quinn Nichol. 2021. [Diffusion models beat gans on image synthesis](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794.
- Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. 2021. [Cogview: Mastering text-to-image generation via transformers](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19822–19835.
- Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. 2023. [Structure and content-guided video synthesis with diffusion models](#). *CoRR*, abs/2302.03011.
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. [Generative adversarial nets](#). In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.
- Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2023. [Prompt-to-prompt image editing with cross-attention control](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. 2022a. [Imagen video: High definition video generation with diffusion models](#). *CoRR*, abs/2210.02303.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. [Denoising diffusion probabilistic models](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Jonathan Ho and Tim Salimans. 2022. [Classifier-free diffusion guidance](#). volume abs/2207.12598.
- Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022b. [Video diffusion models](#). In *NeurIPS*.
- Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. 2023. [Scaling up gans for text-to-image synthesis](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver*,



- BC, Canada, June 17-24, 2023, pages 10124–10134. IEEE.
- Johanna Karras, Aleksander Holynski, Ting-Chun Wang, and Ira Kemelmacher-Shlizerman. 2023. [Dreampose: Fashion image-to-video synthesis via stable diffusion](#).
- Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023. [GLIGEN: open-set grounded text-to-image generation](#). *CoRR*, abs/2301.07093.
- Jun Hao Liew, Hanshu Yan, Daquan Zhou, and Jiashi Feng. 2022. [Magicmix: Semantic mixing with diffusion models](#). *CoRR*, abs/2210.16056.
- Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2022. [Magic3d: High-resolution text-to-3d content creation](#). *CoRR*, abs/2211.10440.
- Shilin Lu, Yanzhu Liu, and Adams Wai-Kin Kong. 2023. [TF-ICON: diffusion-based training-free cross-domain image composition](#). *CoRR*, abs/2307.12493.
- Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. 2023. [Videofusion: Decomposed diffusion models for high-quality video generation](#). *CoRR*, abs/2303.08320.
- Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. 2023. [Dragondiffusion: Enabling drag-style manipulation on diffusion models](#). *CoRR*, abs/2307.02421.
- Alexander Quinn Nichol and Prafulla Dhariwal. 2021. [Improved denoising diffusion probabilistic 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 8162–8171. PMLR.
- Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2022. [GLIDE: towards photorealistic image generation and editing with text-guided diffusion models](#). In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 16784–16804. PMLR.
- Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. 2023. [Zero-shot image-to-image translation](#). In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 11:1–11:11. ACM.
- William Peebles and Saining Xie. 2022. [Scalable diffusion models with transformers](#). *CoRR*, abs/2212.09748.
- Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. [Dreamfusion: Text-to-3d using 2d diffusion](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- 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 *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 8748–8763. PMLR.
- 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.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. [Hierarchical text-conditional image generation with CLIP latents](#).
- 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 *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 8821–8831. PMLR.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. [High-resolution image synthesis with latent diffusion models](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE.
- Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. [Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 22500–22510. IEEE.
- Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. 2022a. [Palette: Image-to-image diffusion models](#). In *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, pages 15:1–15:10. ACM.
- Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho,

- David J. Fleet, and Mohammad Norouzi. 2022b. [Photorealistic text-to-image diffusion models with deep language understanding](#).
- 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](#). In *NeurIPS*.
- Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. 2023. [Make-a-video: Text-to-video generation without text-video data](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. 2023. [Make-it-3d: High-fidelity 3d creation from A single image with diffusion prior](#). *CoRR*, abs/2303.14184.
- 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.
- 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](#). *Trans. Mach. Learn. Res.*, 2022.
- Chuanxia Zheng, Tung-Long Vuong, Jianfei Cai, and Dinh Phung. 2022. [Movq: Modulating quantized vectors for high-fidelity image generation](#). In *NeurIPS*.

## A Additional generation examples



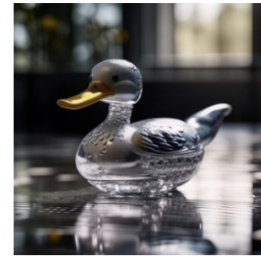
Albert Einstein in a pop art style



A car made of cheese



A corgi knight



Transparent duck made of glass



A bear holding a beer



Sci-fi lady in the outer space



A small cute pig in a glass sphere



An astronaut covered with moss



Schizophrenia



Radiance of consciousness



A man made of newspapers



Three adorable cats in the Universe



Transcendence of the dreams



Mona Lisa as a dog



The best eco city



A red book next to a yellow vase



Cherrypicking scientists



Starship is falling into a black hole



Storm clouds made of marshmallows



A very angry frog

# NewsRecLib: A PyTorch-Lightning Library for Neural News Recommendation

Andreea Iana<sup>1</sup>, Goran Glavas<sup>2</sup>, Heiko Paulheim<sup>1</sup>

<sup>1</sup> Data and Web Science Group, University of Mannheim, Germany

<sup>2</sup> Center For Artificial Intelligence and Data Science, University of Würzburg, Germany

{andreea.iana, heiko.paulheim}@uni-mannheim.de

goran.glavas@uni-wuerzburg.de

## Abstract

NewsRecLib<sup>1</sup> is an open-source library based on Pytorch-Lightning and Hydra developed for training and evaluating neural news recommendation models. The foremost goals of NewsRecLib are to promote *reproducible research* and *rigorous experimental evaluation* by (i) providing a unified and highly configurable framework for exhaustive experimental studies and (ii) enabling a thorough analysis of the performance contribution of different model architecture components and training regimes. NewsRecLib is highly modular, allows specifying experiments in a single configuration file, and includes extensive logging facilities. Moreover, NewsRecLib provides out-of-the-box implementations of several prominent neural models, training methods, standard evaluation benchmarks, and evaluation metrics for news recommendation.

## 1 Introduction

Personalized news recommendation has become ubiquitous for customizing suggestions to users' interests (Li and Wang, 2019; Wu et al., 2023). In recent years, there has been a surge of effort towards neural content-based recommenders. With increasingly complex neural architectures able to ever more precisely capture users' content-based preferences, neural recommenders quickly replaced traditional recommendation models as the go-to paradigm for news recommendation.

Despite the abundance of model designs, research on neural news recommenders (NNRs) suffers from two major shortcomings: (i) a surprising amount of non-reproducible research (Ferrari Dacrema et al., 2021) and (ii) unfair model comparisons (Ferrari Dacrema et al., 2019; Sun et al., 2020). The former is, on the one hand, due to many NNR implementations not being publicly released (Sertkan and Neidhardt, 2022). Existing

open source repositories, on the other hand, expose a multitude of programming languages, libraries, and implementation differences, hindering reproducibility and extensibility (Said and Bellogín, 2014). Moreover, a lack of transparency in terms of evaluation datasets, experimental setup and hyperparameter settings, as well as the adoption of ad-hoc evaluation protocols, further severely impede direct model comparisons. Many personalized news recommenders have been evaluated on proprietary datasets (e.g., Bing News (Wang et al., 2018), MSN News (Wu et al., 2019a,d), News App (Qi et al., 2022)). Even the models trained on the more recently introduced open benchmarks (e.g., Adressa (Gulla et al., 2017), MIND (Wu et al., 2020b)) cannot be directly compared due to the lack of standard dataset splits and evaluation protocols (Wu et al., 2021; Zhang et al., 2021; Gong and Zhu, 2022; Wang et al., 2022). Even more concerning, crucial details regarding the setup of the experiments are regularly omitted from the publications or hard-coded without explanation.

It is thus particularly difficult to evaluate the impact of specific components in NNR architecture and training (e.g., news encoder, user modeling, training objectives) on the overall performance of the model (Iana et al., 2023a). Many models simultaneously change multiple components in both the news and the user encoder, while carrying out only partial ablation studies or evaluating against suboptimal baselines (Rendle et al., 2019).

In this work, we introduce NewsRecLib, an open source library for NNRs, to remedy these critical limitations.<sup>2</sup> NewsRecLib aims to facilitate reproducible research and comprehensive experimental studies, using an end-to-end pipeline powered by a single configuration file that specifies a complete experiment – from dataset selection and preprocessing over model architecture and training to evaluation protocol and metrics. NewsRecLib is

<sup>1</sup><https://github.com/andreeaiana/newsreclib>

<sup>2</sup>The library is licensed under a MIT license.

built based on the following guiding principles:

**Modularity and extensibility.** With PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) as its backbone, NewsReCLib is designed in a modular fashion, with core individual components being decoupled from one another. This enables mixing and matching different modules, as well as seamlessly integrating new ones.

**Easy configurability and reproducibility.** NewsReCLib is powered by Hydra (Yadan, 2019), in which each experiment is defined through a single configuration file composed from the configurations of specific pipeline components. The configuration of every experiment is automatically stored at the start of the run and as such trivially enables reproducibility.

**Logging and profiling.** The library supports multiple standard tools (e.g., WandB (Biewald, 2020), Tensorboard (Abadi et al., 2016)) for extensive logging, monitoring, and profiling of experiments with neural models – in terms of losses, evaluation metrics, runtime, memory usage, and model size.

Overall, NewsReCLib is designed to support the development and benchmarking of NNRs as well as the specific analysis of contributions of common components of the neural recommendation pipelines. In this paper, we discuss the building blocks of NewsReCLib and provide an overview of the readily available models. For a detailed documentation on the usage of the library, we refer to its project page.

## 2 NewsReCLib – the Library

Figure 1 depicts the structure of NewsReCLib, comprising different functional modules: from data modules for downloading and processing datasets to recommendation modules for training and evaluating a particular NNR. The overall pipeline of an experiment is built automatically from the high-level experimental flow provided by the user in the form of a single Hydra configuration file.

### 2.1 Modularization and Extensibility

NewsReCLib is highly modularized: it decouples core components to the largest extent possible. This allows for combinations of different news encoders (e.g., over different input features – text, aspects, entities) with different user modeling techniques, click fusion strategies, and training objectives. NewsReCLib is easily extensible with new

features: the user only needs to write a new sub-component class (e.g., category encoder), or, in the case of new datasets or recommenders, to define a new PyTorch Lightning data module or (model) module, respectively.

Concretely, we decouple the essential building blocks of a NNR, namely the *news encoder* (NE), the *user encoder* (UE), and the *click predictor*. NE is further decomposed into a configurable set of feature encoders (i.e., components that embed different aspects of the news, e.g., title, topical category or named entities). Different model components can be interchanged with corresponding sub-modules of other recommenders, ensuring freedom in choosing each building block of a model independently of the other components (i.e., by mixing the NE of “NNR 1” with the UE of “NNR 2”), in contrast to practices in existing NNR libraries, in which sub-components are tied to concrete NNR architectures that introduced them. Because of this, NewsReCLib allows for clear-cut and comprehensive analyses of impact of NNR components on their overall performance.<sup>3</sup> NewsReCLib currently implements feature encoders used in pre-implemented models (see Appendix §B); users can, however, easily incorporate new ones (e.g., an image encoder) by extending the respective class.

### 2.2 Configurability and Reproducibility

Reproducibility strongly relies on the transparency of each step and component in the pipeline, as well as the availability of metadata regarding the factors that influence the model (e.g., hyperparameter values, training objective) and the environment in which it is trained and evaluated (e.g., library versions). Because of this, NewsReCLib leverages the Hydra<sup>4</sup> framework (Yadan, 2019) to decouple the experiment configuration (i.e., a pipeline of modules) from the concrete implementations (i.e., source code) of the modules.

Each concrete module setting is specified and retrieved automatically from a dedicated configuration file which can be accessed by all the pipeline components. A variety of callbacks supported by PyTorch Lightning (e.g., model checkpointing, early stopping, debugging) can be defined, and modified via a corresponding configuration. A single configuration file guides each experiment:

<sup>3</sup>E.g., we leveraged an earlier version of NewsReCLib to analyze the impact of click behavior fusion strategies and training objectives on NNRs’ performance (Iana et al., 2023a).

<sup>4</sup><https://hydra.cc/>

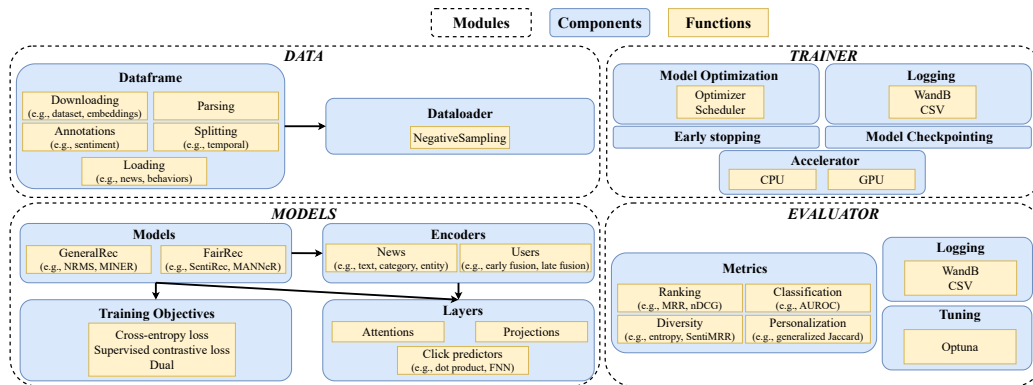


Figure 1: Illustration of the NewsRecLib framework.

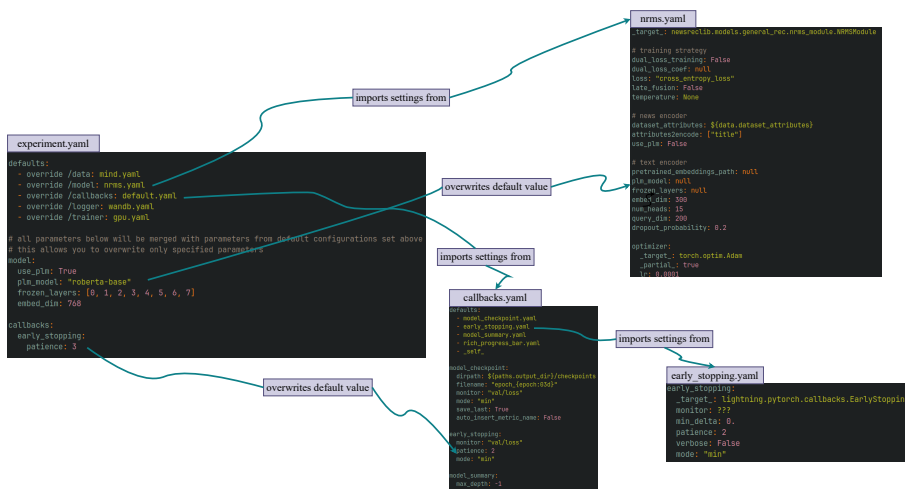


Figure 2: A minimal configuration example for training an NRMS (Wu et al., 2019d) model. All settings defined in the main and the imported configuration files are merged and persisted into a single configuration object.

the default configurations of the used modules and callbacks are hierarchically inherited and can be overridden. Experiment configurations can also be overwritten directly from the command line, removing the need to store many similar configuration files: this facilitates fast experimentation and minimizes boilerplate code. Experiments can be executed on CPU, GPU, and in a distributed fashion by specifying the type of accelerator supported in PyTorch Lightning. The integration with extensive logging capabilities (see §2.3) ensures that any modifications are persistently stored in the experiment directory, together with other log files and model checkpoints.

Fig. 2 shows a minimal configuration example for an experiment that trains an instance of the NRMS (Wu et al., 2019d) model. The main configuration file `experiment.yaml` guides the pipeline. It inherits the data and model-specific configurations from `mind.yaml` and `nrms.yaml`, which specify the default configurations of the data module

and NNR model, respectively. `experiment.yaml` further inherits the default configurations for the WandB logger, the trainer, and various callbacks. The example also illustrates the interplay between modularization and configurability: we replace the original NE of the NRMS model with a pretrained language model (in this case `roberta-base`).

### 2.3 Performance Evaluation and Profiling

With Hydra’s pluggable architecture as its backbone, every part of the recommendation pipeline is transparent to the user. NewsRecLib records comprehensive information during training, including number of trainable model parameters and total model size, runtimes, training and validation losses. Moreover, it stores important metadata regarding hyperparameter settings, operating system, PyTorch version, environment details, and dependencies between libraries. Any profiler supported by PyTorch can be incorporated by a simple modification of the corresponding configuration file.

NewsReclib supports widely used loggers like WandB<sup>5</sup> (Biewald, 2020) and Tensorboard<sup>6</sup> (Abadi et al., 2016). Moreover, users can export evaluation metrics for further analysis. Appendix A shows an example of the logging output. We rely on TorchMetrics<sup>7</sup> (Detlefsen et al., 2022) for model evaluation. Users can track numerous metrics ranging from accuracy-based to beyond-accuracy (e.g., diversity) performance. New metrics can be easily added to the pipeline, either by defining the necessary callbacks in the case of metrics already available in TorchMetrics, or by implementing a custom metric as a subclass of the base Metric class in TorchMetrics.

## 2.4 Hyperparameter Optimization

NNR performance heavily depends on model hyperparameters, making hyperparameter optimization a crucial ingredient in the empirical evaluations of NNRs. NewsReclib supports hyperparameter tuning using the Optuna framework (Akiba et al., 2019), which offers a wide range of samplers, such as random search, grid search, and Bayesian optimization (Bergstra et al., 2011; Ozaki et al., 2020).<sup>8</sup> In conjunction with the modularity of NewsReclib, this allows nearly every component of a news recommender to be treated as a hyperparameter, so that users can optimize the choice of encoders or scoring functions. Figure 3 shows a basic multi-objective hyperparameter search over the number of negative samples, the model’s learning rate, and temperature for the supervised contrastive loss.

## 2.5 Available Modules

NewsReclib currently encompasses two popular benchmark datasets, 13 news recommendation models, and various evaluation metrics.

**Datasets.** We provide out-of-the-box utilities for two prominent monolingual news recommendation benchmarks: MIND (Wu et al., 2020b) (with English news) and Adressa (Gulla et al., 2017) (with Norwegian news). For both datasets, NewsReclib supports automatic downloading (when available)<sup>9</sup>,

<sup>5</sup><https://wandb.ai/site>

<sup>6</sup><https://www.tensorflow.org/tensorboard>

<sup>7</sup><https://torchmetrics.readthedocs.io/en/stable/>

<sup>8</sup><https://optuna.readthedocs.io/en/stable/index.html>

<sup>9</sup>Note that for the Adressa dataset, only a limited version of the dataset is available for download. For the full version containing additional features, users should contact the authors, as detailed in <https://reclab.idi.ntnu.no/dataset/>

```
defaults:
  - override /hydra/sweeper: optuna
optimized_metric: "val/acc_best"
hydra:
  mode: "MULTIRUN"
  sweeper:
    _target_: hydra_plugins.hydra_optuna_sweeper.optuna_sweeper.OptunaSweeper
    storage: null
    study_name: null
    n_jobs: 1
    direction: minimize
    n_trials: 20
    sampler:
      _target_: optuna.samplers.TPESampler
      seed: 1234
      n_startup_trials: 10
    params:
      data_neg_sampling_ratio: range(1, 10, step=1)
      model_lr: choice(1e-4, 1e-5, 1e-6)
      model_temperature: interval(0.8, 1.0)
```

Figure 3: Example of a hyperparameter optimization process. The configuration first runs 10 trials of a search using Bayesian optimization. The hyperparameter search space is defined by indicating the interval, range or choice of values for each desired parameter.

data parsing, and pre-processing functionalities to create a unified PyTorch Lightning datamodule. For both datasets, we include their small and large versions, MINDsmall and MINDlarge, and Adressa-1 week and 10 weeks, respectively.

Since Wu et al. (2020b) do not publicly release test labels for MIND, we use the provided validation portion for testing, and split the respective training set into temporally disjoint training and validation portions. We follow established practices on splitting the Adressa dataset (Hu et al., 2020; Xu et al., 2023) into train, validation, and test sets. In contrast to MIND, which consists of impression log (lists of clicked and non-clicked news by the user), the Adressa dataset contains only positive samples (Gulla et al., 2017). Following Yi et al. (2021), we build impressions by randomly sampling 20 news as negatives for each clicked article.

We additionally automatically annotate datasets with sentiment labels obtained by VADER (Hutto and Gilbert, 2014), a monolingual (English) rule-based algorithm (only for MIND), and a multilingual sentiment classification model of Barbieri et al. (2022), fine-tuned from XLM-RoBERTa Base (Conneau et al., 2020).

**Recommendation Models.** NewsReclib provides implementations for 10 general-purpose NNRs and 3 fairness-aware recommenders. To support analysis of model components, for the models that did not use PLMs in their NEs (but rather contextualized embeddings with convolutional or attention layers), we implement an additional variant with a PLM-based NE (as proposed in Wu et al. (2021)). Furthermore, models can be trained either with *early fusion*, i.e., learning a parameterized user encoder to aggregate embeddings of news or the

simpler *late fusion* strategy proposed in Iana et al. (2023a), which replaces explicit user encoders with parameter-efficient dot products between candidate and clicked news embeddings. Appendix B details all available configurations for each recommendation model.

**Training Objectives.** Most NNR models are trained with point-wise classification objectives (Wang et al., 2018; Wu et al., 2019a,d) with negative sampling (Wu et al., 2019b, 2022a). In Iana et al. (2023a), we have shown that contrastive learning constitutes a viable alternative. At the same time, combining point-wise classification with contrastive objectives has been successfully employed in related tasks (Gunel et al., 2020). We thus implement three training objectives: cross-entropy loss, supervised contrastive loss (Khosla et al., 2020), and a dual objective that is a weighted average between the two.

**Evaluation Metrics.** NewsReCLib integrates standard accuracy-based metrics, such as AUC, MRR, and nDCG@ $k$ . Additionally, we implement aspect-based diversity and aspect-based personalization defined in Iana et al. (2023b). The availability of these beyond-accuracy metrics enables multi-faceted evaluation of NNRs.

### 3 Comparison to Related Frameworks

In the past decade, numerous frameworks for the development and comprehensive evaluation of recommender systems have been proposed to address the problem of reproducibility in the field (Gantner et al., 2011; Ekstrand et al., 2011; Ekstrand, 2020; Guo et al., 2015; Kula, 2017; Da Costa et al., 2018; Salah et al., 2020; Hug, 2020; Sun et al., 2020; Anelli et al., 2021). News recommendation poses different challenges for practitioners in comparison to recommendation in domains such as movies, music, or e-commerce (Raza and Ding, 2022; Iana et al., 2022). However, few of the existing and widely used libraries offer support for news recommenders, and especially for the modern neural news recommendation models.

Microsoft Recommenders (Graham et al., 2019; Argyriou et al., 2020) and RecBole (Zhao et al., 2021, 2022) provide implementation for five and three NNRs, respectively, as well as utilities for the MIND dataset. Nonetheless, other datasets, more recent approaches, and in particular fairness-aware models and beyond-accuracy metrics are not supported. StreamingRec (Jugovac et al., 2018) is a

framework for evaluating streaming-based news recommenders, covering a wide range of algorithms, from trivial baselines (e.g., recently published, most popular) or item-to-item based collaborative filtering or session-based nearest neighbor techniques, to association rule methods and content-based approaches. However, it does not support any of the recent neural models. In these libraries, the sub-modules of a specific recommender are not decoupled from the overall model, which impedes experimentation with and analysis of different model components and training strategies.

In contrast to these frameworks, NewsReCLib focuses solely on the state-of-the-art neural news recommendation models, providing utilities for the most used benchmark datasets, architectures, training techniques, and evaluation metrics tailored to news recommendation. NewsReCLib unifies disparate implementations of recent neural news recommenders in a single open-source library that is built on top of mature frameworks for deep learning (PyTorch Lightning), evaluation (TorchMetrics), and configuration (Hydra).

## 4 Experiments

We conduct experiments with the pre-implemented recommendation models from NewsReCLib to investigate their performance when (1) trained with the original architecture (e.g., NE based on word embeddings and contextualization layer) and (2) trained with a PLM-based NE (Wu et al., 2021).

### 4.1 Datasets and Experimental Setup

We carry out the evaluation on the MINDsmall (Wu et al., 2020b) (denoted MIND) and Adressa-1 week (denoted Adressa) (Gulla et al., 2017) benchmark datasets. We evaluate two versions of the models, namely (1) with the original NE and (2) the NE modified to use a PLM (Wu et al., 2021) (if not used in the original NE). We use RoBERTa Base (Liu et al., 2019) and NB-BERT Base (Kummer-vold et al., 2021; Nielsen, 2023) for experiments on MIND and Adressa, respectively. In both cases, we fine-tune only the last four layers of the PLM in the interest of computational efficiency. We use

<sup>10</sup>We use the LSTUR<sub>ini</sub> version of the model. For details, refer to An et al. (2019).

<sup>11</sup>We use the MINER *weighted* version of the model. For details, refer to Li et al. (2022).

<sup>12</sup>We use the MANNeR version which performs multi-aspect diversification with  $\lambda_{ctg} = -0.15$  and  $\lambda_{snt} = -0.25$  for MIND, and  $\lambda_{ctg} = -0.35$  and  $\lambda_{snt} = -0.25$  for Adressa, respectively. For details, refer to Iana et al. (2023b).



| Model                          | MIND            |                 |                 |                 |                 |                 | Adressa         |                 |                 |                 |                 |                 |
|--------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                                | AUC             | MRR             | nDCG@5          | nDCG@10         | $D_{ctg}$ @10   | $D_{snt}$ @10   | AUC             | MRR             | nDCG@5          | nDCG@10         | $D_{ctg}$ @10   | $D_{snt}$ @10   |
| DKN                            | 50.0±0.0        | 26.3±0.4        | 24.6±0.5        | 31.5±0.3        | 50.4±1.0        | 66.0±0.6        | –               | –               | –               | –               | –               | –               |
| NPA                            | 55.1±0.6        | 28.5±1.1        | 26.4±1.1        | 32.9±1.0        | 51.8±0.2        | 67.5±0.7        | 53.3±3.5        | 31.8±1.9        | 30.4±0.3        | 38.2±2.8        | 31.6±0.3        | 60.7±0.4        |
| NRMS                           | 54.1±0.8        | 27.2±0.6        | 25.3±0.5        | 31.9±0.4        | 52.1±0.6        | 65.9±1.7        | 63.8±4.7        | 30.5±3.6        | 28.6±5.5        | 37.1±4.8        | 31.7±0.3        | 60.7±0.7        |
| NAML                           | 50.2±0.0        | 33.4±0.6        | 31.8±0.7        | 38.1±0.5        | 47.0±1.0        | 66.9±0.3        | 50.0±0.0        | 37.8±3.5        | 38.2±4.1        | 45.1±3.6        | 31.5±4.6        | 60.7±0.4        |
| GeneralRec LSTUR <sup>10</sup> | 58.8±2.1        | 32.2±0.9        | 30.4±0.9        | 36.8±0.9        | 43.1±1.2        | 65.6±0.6        | 68.1±2.4        | <b>38.0±1.7</b> | <b>39.1±2.3</b> | <b>45.9±2.4</b> | 27.7±2.4        | 60.1±0.3        |
| TANR                           | 53.0±4.1        | 30.7±0.6        | 29.0±0.5        | 35.3±0.4        | 50.5±0.4        | 66.7±0.8        | 50.3±0.5        | 33.4±3.4        | 32.9±4.7        | 40.0±4.1        | 29.8±0.9        | 60.1±0.3        |
| CAUM                           | 59.5±0.6        | 33.1±0.4        | 31.2±0.5        | 37.7±0.5        | 47.4±0.5        | 66.7±0.8        | <u>72.5±2.3</u> | 36.0±3.1        | 37.7±4.4        | 44.9±3.1        | 29.3±3.3        | 60.5±0.3        |
| MINS                           | 56.1±1.5        | 31.0±1.5        | 29.4±1.5        | 35.7±1.5        | 47.0±1.7        | 67.6±1.1        | <b>73.8±3.2</b> | <u>37.4±2.5</u> | <u>38.8±4.1</u> | <u>45.8±3.2</u> | 32.4±0.8        | 60.6±0.3        |
| CenNewsRec                     | 54.7±1.3        | 26.9±0.8        | 25.4±0.8        | 32.0±0.7        | 50.9±0.7        | 68.1±0.7        | 62.3±2.1        | 29.3±2.6        | 26.9±3.9        | 35.1±3.2        | 31.7±0.5        | 60.7±0.3        |
| FairRec SentiRec               | 52.0±0.5        | 27.2±0.9        | 25.2±1.0        | 31.8±0.8        | 52.5±1.2        | 67.7±1.1        | 55.0±0.7        | 26.9±0.4        | 24.3±0.7        | 30.1±0.7        | <u>35.2±0.1</u> | <b>66.1±0.7</b> |
| SentiDebias                    | 56.6±1.7        | 25.4±0.7        | 23.7±0.9        | 30.3±0.6        | 53.5±1.3        | 68.1±1.3        | 66.5±0.9        | 29.4±0.7        | 29.2±1.6        | 36.9±1.2        | 31.3±0.8        | 61.1±0.3        |
| NRMS-PLM                       | 50.0±0.0        | 21.9±2.8        | 19.5±2.9        | 26.0±3.0        | 53.2±1.7        | 66.1±3.4        | 53.1±2.7        | 34.9±2.5        | 34.7±3.0        | 42.8±2.8        | 32.3±1.2        | 61.6±0.3        |
| NAML-PLM                       | 52.8±2.4        | 30.0±1.2        | 28.2±1.3        | 34.7±1.2        | 39.3±2.5        | 66.9±0.6        | 50.0±0.0        | 35.3±2.8        | 35.0±3.8        | 41.3±3.7        | 26.7±6.6        | 60.6±0.5        |
| LSTUR-PLM                      | 50.0±0.0        | 30.7±0.6        | 29.0±0.6        | 35.3±0.6        | 36.6±0.9        | 67.0±0.9        | 55.5±2.3        | 30.4±1.6        | 28.8±2.3        | 35.3±2.2        | 22.3±2.8        | 60.9±0.4        |
| GeneralRec TANR-PLM            | 50.0±0.7        | 25.9±3.5        | 23.3±3.6        | 29.8±3.4        | 47.6±6.8        | 61.4±3.0        | 50.0±0.0        | 35.5±3.8        | 35.1±5.1        | 41.8±4.7        | 24.8±10.0       | 59.9±1.0        |
| CAUM-PLM                       | 59.7±2.0        | 32.8±0.5        | 31.0±0.6        | 37.2±0.5        | 44.3±2.2        | 67.5±0.8        | 66.1±2.3        | 30.7±6.8        | 30.6±8.4        | 35.7±9.9        | 22.9±3.4        | 60.4±0.4        |
| MINS-PLM                       | 50.0±0.7        | 22.4±3.5        | 20.2±3.9        | 26.5±4.0        | 50.6±3.2        | 67.3±1.1        | 65.3±4.4        | 33.1±2.8        | 31.5±4.6        | 40.4±3.9        | 26.6±5.5        | 60.5±0.5        |
| CenNewsRec-PLM                 | 50.0±0.2        | 21.2±2.8        | 18.9±2.9        | 25.4±2.8        | 54.2±1.3        | 67.0±1.7        | 54.4±5.3        | 35.8±3.1        | 35.9±3.3        | 42.8±2.1        | 31.6±0.8        | 61.0±0.6        |
| MINER <sup>11</sup>            | 51.2±0.4        | 24.2±0.5        | 22.0±0.6        | 28.2±0.5        | <b>54.8±0.3</b> | 68.8±0.6        | 55.3±6.9        | 33.5±2.2        | 33.1±3.3        | 39.1±3.3        | 32.4±1.4        | 61.2±1.4        |
| FairRec SentiRec-PLM           | 50.0±0.6        | 24.7±0.7        | 22.6±0.6        | 29.1±0.6        | 52.3±2.4        | 67.2±2.1        | 61.2±3.0        | 31.6±3.4        | 30.4±4.4        | 38.2±4.4        | 32.9±1.7        | 59.9±2.4        |
| SentiDebias-PLM                | 51.0±0.5        | 28.7±0.4        | 27.5±0.4        | 34.0±0.4        | 47.7±2.0        | 67.9±1.7        | 67.3±2.8        | 37.1±3.6        | 38.0±5.1        | 45.3±3.8        | 32.6±1.2        | 61.5±1.0        |
| MANNr <sup>12</sup>            | <b>66.2±1.0</b> | <b>36.7±1.3</b> | <b>35.1±1.3</b> | <b>41.1±1.1</b> | 50.5±0.3        | <b>68.2±0.4</b> | 67.6±4.3        | 31.9±2.8        | 30.5±4.1        | 38.9±3.9        | <b>39.2±0.4</b> | <u>64.9±0.5</u> |

Table 1: Recommendation and aspectual diversity (in terms of topical categories  $D_{ctg}$  and sentiments  $D_{snt}$ ) performance of different neural news recommenders. We report averages and standard deviations across five different runs. The best results per column are highlighted in bold, the second best are underlined. The dashed line separates the general (GeneralRec) from the fairness-aware (FairRec) recommendation models.

100-dimensional TransE embeddings (Bordes et al., 2013) pretrained on Wikidata as input to the entity encoder for models using named entities as input features to their NEs, a maximum history length of 50, and set all other model-specific hyperparameters to optimal values reported in the respective papers. We train all models with mixed precision, and optimize with the Adam algorithm (Kingma and Ba, 2014), with the learning rate of 1e-4. We train models with a PLM-empowered NE for 10 epochs, and the model variant without PLMs for 20 epochs. Since Adressa contains no abstract or disambiguated named entities, we use only the title for the models benchmarked on that dataset.

## 4.2 Results

Table 1 summarizes the results on content-based recommendation performance (w.r.t. AUC, MRR, nDCG@5, nDCG@10) and aspect diversification for topical categories ( $D_{ctg}$ ) and sentiment ( $D_{snt}$ ), as per Iana et al. (2023b). We find that PLM-based NEs do not necessarily lead to performance improvements. We hypothesize that this is due to the dataset size: a PLM-based NE requires training a larger number of parameters than one which contextualizes pretrained word embeddings with a CNN or attention network. Note that rather small improvements of PLM-empowered NEs over original NEs have been shown only for larger-scale datasets (Wu et al., 2021). These findings indicate that more research is needed to understand under which settings older NEs can still benefit NNRs. MANNr, with its late click behavior fusion approach, out-

performs all other models on MIND, but it underperforms on Adressa. Note that the contrastive learning training approach adopted by MANNr (Iana et al., 2023b) benefits from larger training datasets, and MINDsmall has roughly five times as many news as Adressa 1-week. Expectedly, w.r.t. aspect-based diversity, NNRs with diversification objectives (e.g., for sentiment) outperform models trained only to maximize content-based accuracy.

## 5 Conclusion

In this work, we introduced NewsReLib, a highly configurable, modular and easily extensible framework for neural news recommendation. Our library is specifically designed to foster reproducible research in recommender systems and rigorous evaluation of models – users only need to create a single configuration file for an experiment. We briefly described the underlying principles of NewsReLib and the structure of its building blocks. The framework currently provides two standard benchmark datasets, loading and pre-processing functions, 13 neural recommendation models, different training objectives and hyperparameters optimization strategies, numerous evaluation metrics, extensive logging capabilities, and GPU support. We believe that NewsReLib is a useful tool for the community that will (i) catalyze reproducible NNR research, (ii) foster fairer comparisons between the models, and (iii) facilitate identification of NNR components that drive their performance.

## Limitations

While we have striven to build a comprehensive library for the design and fair evaluation of neural news recommendation models, several additional factors must be taken into account. Firstly, even though we aim to replicate the original implementations of the models to the highest degree possible, discrepancies in our code and results can arise from the usage of different frameworks, as well as scarce availability of implementation details in the source code or publications of some of the recommenders. Secondly, our library is heavily dependent on the changes and maintenance of the frameworks on which it is built, namely PyTorch Lightning (and by extension, PyTorch), Hydra, TorchMetrics, Optuna. As such, new plugins for logging (e.g., Neptune (Neptune team, 2019), Comet (Rei et al., 2020), MLFlow (Zaharia et al., 2018)) or hyperparameter optimization (e.g., Ax<sup>13</sup>) need to be integrated with PyTorch Lightning and Hydra.

Moreover, we rely on open benchmark news datasets for training and evaluating the recommenders. Consequently, any biases that might be contained in the news and user data could be propagated through the recommendation pipeline. Additionally, the usage of these datasets is intertwined with their public availability. Any changes to the datasets or access restrictions are likely to impact the way pre-implemented models in NewsReclib can be trained and benchmarked.

Lastly, neural news recommendation is a computationally expensive endeavor which requires availability of large compute resources. Although NewsReclib technically supports execution of experiments on CPU, this would be not only highly inefficient and time-consuming, but also infeasible for large-scale datasets with hundreds of thousands of users and news. Consequently, users should ideally have access to GPUs to efficiently use our library.

## Ethics Statement

Users of our library should differentiate the recommendation models available in NewsReclib from the originals. Consequently, they should explicitly credit and cite both NewsReclib, as well as the original implementations, as specified on our GitHub page.

<sup>13</sup><https://ax.dev/>

## Acknowledgements

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG.

## References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.
- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 336–345.
- Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2405–2414.
- Andreas Argyriou, Miguel González-Fierro, and Le Zhang. 2020. Microsoft recommenders: Best practices for production-ready recommendation systems. In *Companion Proceedings of the Web Conference 2020*, pages 50–51.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.
- Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. XLM-T: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 258–266.
- James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

- Lukas Biewald. 2020. [Experiment tracking with weights and biases](#). Software available from wandb.com.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.
- Arthur Da Costa, Eduardo Fressato, Fernando Neto, Marcelo Manzato, and Ricardo Campello. 2018. Case recommender: a flexible and extensible python framework for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 494–495.
- Nicki Skafté Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. 2022. [Torchmetrics - measuring reproducibility in pytorch](#).
- 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.
- Michael D Ekstrand. 2020. Lenskit for python: Next-generation software for recommender systems experiments. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 2999–3006.
- Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 133–140.
- William Falcon and The PyTorch Lightning team. 2019. [PyTorch Lightning](#).
- Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, 39(2):1–49.
- Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, pages 101–109.
- Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308.
- Shansan Gong and Kenny Q Zhu. 2022. Positive, negative and neutral: Modeling implicit feedback in session-based news recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1185–1195.
- Scott Graham, Jun-Ki Min, and Tao Wu. 2019. Microsoft recommenders: tools to accelerate developing recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 542–543.
- Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The adressa dataset for news recommendation. In *Proceedings of the international conference on web intelligence*, pages 1042–1048.
- Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.
- Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. Librec: A java library for recommender systems. In *UMAP workshops*, volume 4, pages 38–45. Citeseer.
- Linmei Hu, Siyong Xu, Chen Li, Cheng Yang, Chuan Shi, Nan Duan, Xing Xie, and Ming Zhou. 2020. Graph neural news recommendation with unsupervised preference disentanglement. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4255–4264.
- Nicolas Hug. 2020. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174.
- 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.

- Andreea Iana, Mehwish Alam, and Heiko Paulheim. 2022. A survey on knowledge-aware news recommender systems. *Semantic Web*, (Preprint):1–62.
- Andreea Iana, Goran Glavas, and Heiko Paulheim. 2023a. Simplifying content-based neural news recommendation: On user modeling and training objectives. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2384–2388.
- Andreea Iana, Goran Glavaš, and Heiko Paulheim. 2023b. Train once, use flexibly: A modular framework for multi-aspect neural news recommendation. *arXiv preprint arXiv:2307.16089*.
- Michael Jugovac, Dietmar Jannach, and Mozghan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM conference on recommender systems*, pages 269–273.
- Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschiot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Maciej Kula. 2017. Spotlight. <https://github.com/maciejkula/spotlight>.
- Per E Kummervold, Javier De La Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. 2021. Operationalizing a national digital library: The case for a norwegian transformer model. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 20–29.
- Jian Li, Jieming Zhu, Qiwei Bi, Guohao Cai, Lifeng Shang, Zhenhua Dong, Xin Jiang, and Qun Liu. 2022. MINER: multi-interest matching network for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 343–352.
- Miaomiao Li and Licheng Wang. 2019. A survey on personalized news recommendation technology. *IEEE Access*, 7:145861–145879.
- 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*.
- Neptune team. 2019. [neptune.ai](https://neptune.ai).
- Dan Nielsen. 2023. *ScandEval: A benchmark for Scandinavian natural language processing*. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 185–201, Tórshavn, Faroe Islands. University of Tartu Library.
- Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. 2020. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pages 533–541.
- Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. 2022. News recommendation with candidate-aware user modeling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1917–1921.
- Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-preserving news recommendation model learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1423–1432.
- Shaina Raza and Chen Ding. 2022. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review*, pages 1–52.
- Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*.
- Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395*.
- Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136.
- Aghiles Salah, Quoc-Tuan Truong, and Hady W Lauw. 2020. Cornac: A comparative framework for multimodal recommender systems. *The Journal of Machine Learning Research*, 21(1):3803–3807.
- Mete Sertkan and Julia Neidhardt. 2022. Diversifying sentiments in news recommendation. In *Proceedings of the 2nd Perspectives on the Evaluation of Recommender Systems Workshop*.
- Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 23–32.
- 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.
- Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844.
- Rongyao Wang, Shoujin Wang, Wenpeng Lu, and Xueping Peng. 2022. News recommendation via multi-interest news sequence modelling. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7942–7946. IEEE.
- Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019a. Neural news recommendation with attentive multi-view learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3863–3869.
- Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019b. NPA: neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2576–2584.
- Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. 2019c. Neural news recommendation with topic-aware news representation. In *Proceedings of the 57th Annual meeting of the association for computational linguistics*, pages 1154–1159.
- Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019d. Neural news recommendation with multi-head self-attention. 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 6389–6394.
- Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2022a. Rethinking InfoNCE: How many negative samples do you need? In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence ((IJCAI-22))*, pages 2509–2515.
- Chuhan Wu, Fangzhao Wu, Yongfeng Huang, and Xing Xie. 2023. Personalized news recommendation: Methods and challenges. *ACM Transactions on Information Systems*, 41(1):1–50.
- Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020a. Sentirec: Sentiment diversity-aware neural news recommendation. 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 44–53.
- Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1652–1656.
- Chuhan Wu, Fangzhao Wu, Tao Qi, Wei-Qiang Zhang, Xing Xie, and Yongfeng Huang. 2022b. Removing ai’s sentiment manipulation of personalized news delivery. *Humanities and Social Sciences Communications*, 9(1):1–9.
- Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020b. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606.
- Hongyan Xu, Qiyao Peng, Hongtao Liu, Yueheng Sun, and Wenjun Wang. 2023. Group-based personalized news recommendation with long-and short-term fine-grained matching. *ACM Transactions on Information Systems*.
- Omry Yadan. 2019. *Hydra - a framework for elegantly configuring complex applications*. Github.
- Jingwei Yi, Fangzhao Wu, Chuhan Wu, Ruixuan Liu, Guangzhong Sun, and Xing Xie. 2021. Efficient-FedRec: Efficient federated learning framework for privacy-preserving news recommendation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2824.
- Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45.
- Qi Zhang, Qinglin Jia, Chuyuan Wang, Jingjie Li, Zhaowei Wang, and Xiuqiang He. 2021. Amm: Attentive multi-field matching for news recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1588–1592.
- Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. 2022. Recbole 2.0: towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4722–4726.
- Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, pages 4653–4664.

## A Logging

### A.1 Configuration Logging

Figs. 4 and 5 illustrate an example of how the configuration of each of the pipeline’s components is logged when the training process is initiated.

### A.2 Model Metadata Logging

Fig. 6 shows an example of logging relevant metadata information regarding a model’s size and number of parameters.

## B Supported Recommendation Models and Configurations

NewsRecLib provides, to date, implementations of 10 general NNRs:

- *DKN* (Wang et al., 2018) uses a word-entity aligned knowledge-aware convolutional neural network (CNN) (Kim, 2014) to produce news embeddings. It learns candidate-aware representations of users as the weighted sum of their clicked news embeddings, where the weights are computed by an attention network that takes as input the embeddings of the candidate and of the clicked news.
- *NPA* (Wu et al., 2019b) contextualizes pre-trained word embeddings with a CNN, followed by a personalized attention module. Its UE consists of a similar personalized attention module which aggregates the representations of the users’ clicked news, with projected embeddings of the users IDs as attention queries.
- *NAML* (Wu et al., 2019a) uses a sequence of CNN and additive attention (Bahdanau et al., 2015) to contextualize pretrained word embeddings in its NE. Additionally, it leverages category information, with categories embedded through a linear layer. User representations are learned from the embeddings of users’ clicked news with another additive attention layer.
- *NRMS* (Wu et al., 2019d) learns news representations from pretrained word embeddings and a combination of multi-head self-attention (Vaswani et al., 2017) and additive attention; it embeds users with a two-layer encoder consisting also of multi-head self-attention and additive attention.
- *LSTUR* (An et al., 2019) embeds news similarly to NAML (Wu et al., 2019a). However, it learns user representations via recurrent neural networks: it produces short-term user embeddings from the clicked news with a GRU (Cho et al., 2014), which it combines with a long-term embedding, consisting of a randomly initialized and fine-tuned part.
- *TANR* (Wu et al., 2019c) injects information on topical categories, by jointly optimizing the NNR for content personalization and topic classification. It uses the same UE and NE architecture as NAML (Wu et al., 2019a), but does not embed categories.
- *CAUM* (Qi et al., 2022) uses a NRMS-based NE, and additionally encodes title entities with attention layers. Moreover, its candidate-aware UE combines a candidate-aware self-attention network which models long-range dependencies between clicked news, conditioned on the candidate, with a candidate-aware CNN that captures short-term user interests from adjacent clicks, again conditioned on the candidate’s content.
- *MINS* (Wang et al., 2022) embeds textual features of news (i.e., title, abstract) in the same manner as NRMS (Wu et al., 2019d), and categories through a linear embedding layer. Moreover, it uses a combination of multi-head self-attention, multi-channel GRU-based recurrent network, and additive attention to encode users.
- *CenNewsRec* (Qi et al., 2020) combines a CNN network with multi-head self-attention and additive attention modules to produce contextualized representations of news. Its UE resembles that of LSTUR (An et al., 2019), but it learns long-term user vectors from clicked news using a sequence of multi-head self-attention and attentive pooling networks, as opposed to storing an explicit embedding per user.
- *MINER* (Li et al., 2022) uses a pretrained BERT (Devlin et al., 2019) model as NE. Its UE learns multiple user representation vectors using a poly attention scheme that extracts interests vectors through additive attention layers.

| Model                          | News Encoder                  |                             | Click Behavior Fusion |    | Training Objective |     |      |
|--------------------------------|-------------------------------|-----------------------------|-----------------------|----|--------------------|-----|------|
|                                | Word emb. + contextualization | PLM                         | EF                    | LF | CE                 | SCL | Dual |
| GeneralRec                     | DKN (Wang et al., 2018)       | ✓                           | ✗                     | ✓  | ✓                  | ✓   | ✓    |
|                                | NPA (Wu et al., 2019b)        | ✓                           | ✗                     | ✓  | ✓                  | ✓   | ✓    |
|                                | NRMS (Wu et al., 2019d)       | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | NAML (Wu et al., 2019a)       | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | LSTUR (An et al., 2019)       | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | TANR (Wu et al., 2019c)       | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | CAUM (Qi et al., 2022)        | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | MINS (Wang et al., 2022)      | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | CenNewsRec (Qi et al., 2020)  | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | MINER (Li et al., 2022)       | ✓                           | ✓                     | ✓  | ✓                  | ✓   | ✓    |
|                                | FairRec                       | SentiRec (Wu et al., 2020a) | ✓                     | ✓  | ✓                  | ✓   | ✓    |
| SentiDebias (Wu et al., 2022b) |                               | ✓                           | ✓                     | ✓  | ✓                  | ✗   | ✗    |
| MANNeR (Iana et al., 2023b)    |                               | ✓                           | ✓                     | ✓  | ✓                  | ✗   | ✗    |

Table 2: List of currently available models in NewsRecLib, and supported configurations. For *click behavior fusion* we differentiate between early fusion (EF) and late fusion (LF). Models can be trained with cross-entropy loss (CE), supervised contrastive loss (SCL), and a dual objective combining both CE and SCL losses as weighted average (Dual). The dashed line separates the general (GeneralRec) from the fairness-aware (FairRec) recommendation models.

Additionally, NewsRecLib integrates 3 fairness-aware models, namely NNRs that target diversity of recommendations along with pure content-based personalization:

- *SentiRec* (Wu et al., 2020a) uses a similar architecture to NRMS (Wu et al., 2019d) and injects sentiment information by optimizing simultaneously for content personalization, as well as sentiment prediction. Additionally, it regularizes the NNR for sentiment diversity.
- *SentiDebias* (Wu et al., 2022b) is a framework for sentiment debiasing which uses the architecture of NRMS (Wu et al., 2019d), as well as adversarial learning to reduce the model’s sentiment bias (originating from the user data) and generate sentiment-agnostic and diverse recommendations.
- *MANNeR* (Iana et al., 2023b) is a modular framework for multi-aspect neural news recommendation, which comprises two types of modules, each with a corresponding NE (which combines a PLM-based text encoder with an entity embedder consisting of a pretrained embedding and multi-head self-attention layer), which are responsible for content-based, and respectively, aspect-based personalization. Both modules are trained with a contrastive metric objective. MANNeR uses late fusion (Iana et al., 2023a) instead of standard user encoders. At inference time, the

aspect-specific similarity scores are arbitrarily aggregated depending on the downstream task (e.g., content-based personalization, aspect-based diversification) to produce a final ranking of the news.

Table 2 provides an overview of the supported configurations for the available models. For each model, users can choose the type of news encoder, click behavior fusion, and training objective. Note that for some models, due to the high interdependencies between NE and UE, it is not possible to easily replace the original NE with a PLM-based one without breaking the framework’s modularity. Similarly, some models have been designed from the start with a PLM-based NE. In both of these cases, we only provide support for the original NE. Due to the design of some model architectures, changing the training objective would modify the functionality of the model (e.g., using different loss functions in the CR-Module and A-Module of MANNeR (Iana et al., 2023b)). In these cases, we only provide support for one training objective.

```

CONFIG
├── data
│   ├── target : newsreclib.data.mind_rec_datamodule.MINDRecDataModule
│   ├── dataset_size: small
│   ├── dataset_url:
│   │   ├── target:
│   │   │   ├── train: https://mind201910small.blob.core.windows.net/release/MINDlarge_train.zip
│   │   │   ├── dev: https://mind201910small.blob.core.windows.net/release/MINDlarge_dev.zip
│   │   │   └── small:
│   │   │       ├── train: https://mind201910small.blob.core.windows.net/release/MINDsmall_train.zip
│   │   │       ├── dev: https://mind201910small.blob.core.windows.net/release/MINDsmall_dev.zip
│   │   └── pretrained_embeddings_path: https://nlp.stanford.edu/data/glove.840b.300d.zip
│   ├── data_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/data/
│   ├── word_embeddings_dirname: glove
│   ├── word_embeddings_fpath: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/data/glove/glove.840b.300d.txt
│   ├── entity_embeddings_filename: entity_embedding.vec
│   ├── dataset_attributes:
│   │   ├── - title
│   │   ├── - abstract
│   │   ├── - category
│   │   ├── - subcategory
│   │   ├── - title_entities
│   │   ├── - abstract_entities
│   │   ├── - category_class
│   │   ├── - subcategory_class
│   │   └── - sentiment_class
│   ├── id2index_files:
│   │   ├── word2index: word2index.tsv
│   │   ├── entity2index: entity2index.tsv
│   │   ├── categ2index: categ2index.tsv
│   │   ├── subcateg2index: subcateg2index.tsv
│   │   ├── sentiment2index: sentiment2index.tsv
│   │   └── uid2index: uid2index.tsv
│   ├── use_plt: false
│   ├── use_pretrained_categ_embeddings: true
│   ├── word_embed_dim: 300
│   ├── categ_embed_dim: 300
│   ├── entity_embed_dim: 100
│   ├── entity_freq_threshold: 2
│   ├── entity_conf_threshold: 0.5
│   ├── sentiment_annotator:
│   │   ├── target : newsreclib.data.components.sentiment_annotator.BERTSentimentAnnotator
│   │   ├── model_name: cardiffnlp/twitter-xml-roberta-base-sentiment
│   │   ├── tokenizer_max_len: 96
│   │   ├── valid_time_split: '2019-11-14 00:00:00'
│   │   ├── nng_sampling_ratio: 4
│   │   ├── max_title_len: 30
│   │   ├── max_abstract_len: 50
│   │   ├── max_history_len: 50
│   │   ├── concatenate_inputs: false
│   │   ├── tokenizer_name: null
│   │   ├── tokenizer_use_fast: null
│   │   └── tokenizer_max_len: null
│   └── batch_size: 64

```

(a) Data module configuration.

```

model
├── target : newsreclib.models.general_rec_nrms_module.NRMSModule
├── dual_loss_training: false
├── dual_loss_coef: null
├── loss: cross_entropy_loss
├── late_fusion: false
├── temperature: None
├── dataset_attributes:
│   ├── - title
│   ├── - abstract
│   ├── - category
│   ├── - subcategory
│   ├── - title_entities
│   ├── - abstract_entities
│   ├── - category_class
│   ├── - subcategory_class
│   ├── - sentiment_class
│   ├── - sentiment_score
│   └── attributes2encode:
│       ├── - title
│       └── use_plt: false
├── pretrained_embeddings_path: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/data/MINDsmall_train/transformed_word_embeddings.npy
├── plt_model: null
├── freeze_layers: null
├── embed_dim: 300
├── num_heads: 15
├── query_dim: 200
├── dropout_probability: 0.2
├── output:
│   ├── train:
│   │   ├── - preds
│   │   ├── - targets
│   │   └── - cand_news_size
│   ├── val:
│   │   ├── - preds
│   │   ├── - targets
│   │   └── - cand_news_size
│   ├── test:
│   │   ├── - preds
│   │   ├── - targets
│   │   ├── - cand_news_size
│   │   ├── - hist_news_size
│   │   ├── - target_categories
│   │   ├── - target_sentiments
│   │   └── - hist_categories
│   └── - hist_sentiments
├── num_categ_classes: 18
├── num_sent_classes: 3
├── optimizer:
│   ├── target : torch.optim.Adam
│   ├── partial: true
│   ├── lr: 0.0001
│   └── scheduler: null

```

(b) Recommendation module configuration.

Figure 4: Example for logging the configurations of the data and the recommendation modules.



```

callbacks
├── model_checkpoint:
│   ├── target: lightning.pytorch.callbacks.ModelCheckpoint
│   ├── dir_path: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-13-45/checkpoints
│   ├── filename: epoch {epoch:03d}
│   ├── monitor: val/loss
│   ├── verbose: false
│   ├── save_last: true
│   ├── save_top_k: 1
│   ├── mode: min
│   ├── auto_insert_metric_name: false
│   ├── save_weights_only: false
│   ├── every_n_train_steps: null
│   ├── train_time_interval: null
│   ├── every_n_epochs: null
│   ├── save_on_train_epoch_end: null
│   └── early_stopping:
│       ├── target: lightning.pytorch.callbacks.EarlyStopping
│       ├── monitor: val/loss
│       ├── min_delta: 0.0
│       ├── patience: 5
│       ├── verbose: false
│       ├── mode: max
│       ├── strict: true
│       ├── check_finite: true
│       ├── stopping_threshold: null
│       ├── divergence_threshold: null
│       └── check_on_train_epoch_end: null
│   └── model_summary:
│       ├── target: lightning.pytorch.callbacks.RichModelSummary
│       ├── max_depth: -1
│       └── rich_progress_bar:
│           └── target: lightning.pytorch.callbacks.RichProgressBar

```

(a) Callbacks configuration.

```

logger
├── csv:
│   ├── target: lightning.pytorch.loggers.csv_logs.CSVLogger
│   ├── save_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
│   ├── name: csv/
│   └── prefix: ''
│   └── wandb:
│       ├── target: lightning.pytorch.loggers.wandb.WandbLogger
│       ├── save_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
│       ├── offline: false
│       ├── id: null
│       ├── anonymous: null
│       ├── project: newsreclib
│       ├── log_model: false
│       ├── prefix: ''
│       ├── group: mind
│       ├── tags:
│       │   ├── - nrms
│       │   ├── - mindsmall
│       │   ├── - pretrainedemb
│       │   ├── - celoss
│       │   └── - bertsent
│       ├── job_type: ''
│       └── name: nrms_mindsmall_pretrainedemb_celoss_bertsent_s42

```

(b) Logger configuration.

```

trainer
├── target: lightning.pytorch.trainer.Trainer
├── default_root_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
├── min_epochs: 1
├── max_epochs: 20
├── accelerator: gpu
├── devices: 1
├── precision: 16
├── check_val_every_n_epoch: 1
├── deterministic: false
└── profiler: simple

```

(c) Trainer configuration.

```

paths
├── root_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib
├── data_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/data/
├── log_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/
├── output_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib/logs/train/runs/nrms_mindsmall_pretrainedemb_celoss_bertsent_s42/2023-07-30_19-28-23
└── work_dir: /pfs/data5/home/ma_ma/ma_aiana/newsreclib

```

(d) Paths configuration.

Figure 5: Example for logging the configurations of the callbacks, loggers, and trainer.

|    | Name  | Type                            | Params |
|----|---|---------------------------------|--------|
| 0  | train_loss  | MeanMetric                      | 0      |
| 1  | val_loss  | MeanMetric                      | 0      |
| 2  | test_loss   | MeanMetric                      | 0      |
| 3  | val_loss_best   | MinMetric                       | 0      |
| 4  | criterion   | CrossEntropyLoss                | 0      |
| 5  | news_encoder  | NewsEncoder                     | 16.9 M |
| 6  | news_encoder.text_encoders                                    | ModuleDict                      | 16.9 M |
| 7  | news_encoder.text_encoders.title                              | MHSAAddAtt                      | 16.9 M |
| 8  | news_encoder.text_encoders.title.embedding_layer              | Embedding                       | 16.5 M |
| 9  | news_encoder.text_encoders.title.multihead_attention          | MultiheadAttention              | 361 K  |
| 10 | news_encoder.text_encoders.title.multihead_attention.out_proj | NonDynamicallyQuantizableLinear | 90.3 K |
| 11 | news_encoder.text_encoders.title.additive_attention           | AdditiveAttention               | 60.4 K |
| 12 | news_encoder.text_encoders.title.additive_attention.linear    | Linear                          | 60.2 K |
| 13 | news_encoder.text_encoders.title.dropout                      | Dropout                         | 0      |
| 14 | user_encoder  | UserEncoder                     | 421 K  |
| 15 | user_encoder.multihead_attention                              | MultiheadAttention              | 361 K  |
| 16 | user_encoder.multihead_attention.out_proj                     | NonDynamicallyQuantizableLinear | 90.3 K |
| 17 | user_encoder.additive_attention                               | AdditiveAttention               | 60.4 K |
| 18 | user_encoder.additive_attention.linear                        | Linear                          | 60.2 K |
| 19 | click_predictor   | DotProduct                      | 0      |
| 20 | train_rec_metrics   | MetricCollection                | 0      |
| 21 | train_rec_metrics.auc   | BinaryAUROC                     | 0      |
| 22 | train_rec_metrics.mrr   | RetrievalMRR                    | 0      |
| 23 | train_rec_metrics.ndcg@10                                     | RetrievalNormalizedDCG          | 0      |
| 24 | train_rec_metrics.ndcg@5                                      | RetrievalNormalizedDCG          | 0      |
| 25 | val_rec_metrics   | MetricCollection                | 0      |
| 26 | val_rec_metrics.auc   | BinaryAUROC                     | 0      |
| 27 | val_rec_metrics.mrr   | RetrievalMRR                    | 0      |
| 28 | val_rec_metrics.ndcg@10                                       | RetrievalNormalizedDCG          | 0      |
| 29 | val_rec_metrics.ndcg@5  | RetrievalNormalizedDCG          | 0      |
| 30 | test_rec_metrics  | MetricCollection                | 0      |
| 31 | test_rec_metrics.auc  | BinaryAUROC                     | 0      |
| 32 | test_rec_metrics.mrr  | RetrievalMRR                    | 0      |
| 33 | test_rec_metrics.ndcg@10                                      | RetrievalNormalizedDCG          | 0      |
| 34 | test_rec_metrics.ndcg@5                                       | RetrievalNormalizedDCG          | 0      |
| 35 | test_categ_div_metrics  | MetricCollection                | 0      |
| 36 | test_categ_div_metrics.categ_div@10                           | Diversity                       | 0      |
| 37 | test_categ_div_metrics.categ_div@5                            | Diversity                       | 0      |
| 38 | test_sent_div_metrics   | MetricCollection                | 0      |
| 39 | test_sent_div_metrics.sent_div@10                             | Diversity                       | 0      |
| 40 | test_sent_div_metrics.sent_div@5                              | Diversity                       | 0      |
| 41 | test_categ_pers_metrics                                       | MetricCollection                | 0      |
| 42 | test_categ_pers_metrics.categ_pers@10                         | Personalization                 | 0      |
| 43 | test_categ_pers_metrics.categ_pers@5                          | Personalization                 | 0      |
| 44 | test_sent_pers_metrics  | MetricCollection                | 0      |
| 45 | test_sent_pers_metrics.sent_pers@10                           | Personalization                 | 0      |
| 46 | test_sent_pers_metrics.sent_pers@5                            | Personalization                 | 0      |

**Trainable params:** 17.4 M  
**Non-trainable params:** 0  
**Total params:** 17.4 M  
**Total estimated model params size (MB):** 69

Figure 6: Example for logging the model size, number of trainable and non-trainable model parameters.

# MiniChain: A Small Library for Coding with Large Language Models

Alexander M. Rush

srush.research@gmail.com

Hugging Face

Cornell Tech

## Abstract

Programming augmented by large language models (LLMs) opens up many new application areas, but also requires care. LLMs are accurate enough, on average, to replace core functionality, yet make basic mistakes that demonstrate a lack of robustness. An ecosystem of prompting tools, from intelligent agents to new programming languages, has emerged with different solutions for patching LLMs with other tools. In this work, we introduce *MiniChain*, an opinionated tool for LLM augmented programming, with the design goals of ease-of-use of prototyping, transparency through automatic visualization, and a minimalistic approach to advanced features. The *MiniChain* library provides core primitives for coding LLM calls, separating out prompt templates, and capturing program structure. The library includes demo implementations of the main applications papers in the area, including chat-bots, code generation, retrieval-based question answering, and complex information extraction. The library is open-source and available at <https://github.com/srush/MiniChain>, with code demos available at <https://srush-minichain.hf.space/>, and video demo at <https://www.youtube.com/watch?v=VsZ1Vn07sk>.

## 1 Introduction

Large language models (LLMs) (Brown et al., 2020) are a transformative technology that make it possible to develop novel AI applications. Out of the box they perform extremely well across many different domains including code generation, question answering and decomposition, fact retrieval, information extraction, and dialogue to name a few, as well as entirely novel task domains. However, while demonstrating these novel behaviors, they also struggle in basic areas such as mathematical reasoning (Hendrycks et al., 2021), code execution (Liu, 2022), specific document lookup (Guu et al., 2020), and handling long contexts (Shaham et al., 2022).

The gap between the novel general-purpose abilities and low-level deficiencies in known areas, has motivated significant research into multi-stage systems, colloquially *chains*, that combine the use of LLMs with more basic computation blocks and calls to other classical tools. This intermediary software ecosystem describes compositional structure for how the LLM interacts with the scaffolding around it.

Despite the agreed upon problem, there are many different approaches being pursued simultaneously in the open-source community. Systems like AutoGPT (AutoGPT) utilize a fully autonomous agent to direct the choice of supplementary tooling. Other systems like LMQL (Beurer-Kellner et al., 2023) propose a new query language that is able to fully guide and constrict the LLM. In between, the popular LangChain (Chase, 2022) software provides a full-service toolkit for working with many of the different paradigms, from agents to vector-databases to chat bots with memory. The diversity of these systems indicates a broad and open challenge in designing tools that best facilitate programmer interaction with LLMs.

In this white paper, we propose a system with a different goal along this design space. *MiniChain*, is an implementation of the prompt chaining paradigm that can support widely used chaining patterns while remaining significantly simpler than comparable libraries. *MiniChain* is designed with three goals: a) ease-of-use, it should be indistinguishable from standard python code, b) transparency, it should be trivial for the user to introspect and follow all AI calls, c) minimal, it should not implement features that can be done easily with code.

The library itself and the underlying visualization tool are written in python and follow standard coding conventions. The library aims to be understandable by researchers and is contained in one file. However, it is also meant to be complete, and

be possible to implement contemporary research on the topic. To demonstrate this, the whitepaper comes with an implementation of recent prompting research at <https://srush-minichain.hf.space/>. In addition there is a video demo describing its use at <https://www.youtube.com/watch?v=VsZ1Vn07sk>.

## 2 Related Work

Programming with interleaved LLM calls is a very recent phenomenon, and so there are relatively comparable systems. Of the related systems, many exist as open-source libraries or as demo code, and it is difficult to categorize their evolving features. Roughly, prompt chaining systems can be divided into five groups:

*Toolkits for calling LLMs and managing state.* Of these the most representative and important is LangChain (Chase, 2022), a Python library for building LLM applications supporting multiple paradigms including explicit chaining, agent-based modeling, and vector lookups. Dust (Dust) proposes a different toolkit approach using Rust as a backend.

*Programming languages and domain specific languages (DSLs) to support programming with prompts.* These include LMQL (Beurer-Kellner et al., 2023) a DSL for constraining model output, Microsoft’s Semantic Kernel (Microsoft, a), a heavy-weight toolkit supporting many different prompting paradigm across languages, and Demonstrate-Search-Predict (Khattab et al., 2022) a DSL describing systems that integrate retrieval and LLM decision making.

*Collections of tools designed for LLM usage.* Llama-index (Liu, 2022) is a collection of data resources and software meant to help LLMs respond to targeted queries. Other approaches focus on collecting additional models to consult, e.g. HuggingGPT/JARVIS (Shen et al., 2023), or open APIs to utilize as in Taskmatrix (Liang et al., 2023).

*LLM toolkits designed to provide prompts for specific tasks.* These libraries, such as Promptify (Pal, 2022), collect good versions of prompts that help solve specific zero-shot or few-shot tasks. These toolkits are less about the chaining component, but provide clear and usable prompt templates for the individual prompts. Many of the other libraries also provide clear prompts as part of their system.

*Autonomous agents with prompt-supported tools.*

The goal of these systems is less to be integrated into software, and more to propose a different, (and more chaotic) way to solve specific problems through repeated prompting to determine and solve subtasks with external tools. AutoGPT and BabyAGI (AutoGPT; Nakajima) are the most well-known systems in this category.

Of these systems, MiniChain can be seen as fitting between the first and second category. It is an embedded domain specific language in Python with a minimal toolkit supporting common prompt-chaining paradigms.

## 3 Programming with LLMs

Let us begin by considering a practical example of chaining language models. Large language models have trouble with computing mathematical equations due to the limitations of fixed depth transformer models. For example, at the time of this writing asking Google Bard <sup>1</sup> to “sum the numbers 10 through 15” yields a confident assertion that the answer is 60.

However, researchers have noted that they are extremely good at code generation, and can map natural language descriptions of math problems into usable code, e.g. (Gao et al., 2023). This code can then be run to produce an answer. This motivates the base case for *chaining*. Given a word problem, we a) describe to the LLM what we want to do and have it convert it to code, and then b) run the code in an interpreter to produce the result. We will think of both of these steps as “prompting”, i.e. first prompt the LLM and then prompt the interpreter.

The *MiniChain* API has the user describe both of these steps using simple functions. First we describe how to ask to map the problem to code.

```
@prompt(OpenAI(),
        template_file="math.pmpmt.tpl")
def math_prompt(model, query):
    "Call GPT with a Jinja template"
    return model(dict(question=query))
```

The function takes two arguments, a special `model` argument representing the LLM and a user argument for the query to convert to code. The key additional component is the `@prompt` decorator which tells us which model to use (in this case the default OpenAI model), as well as a template file with the details of the prompt to use.

<sup>1</sup><https://bard.google.com/u/2/>

Now let's look at the prompt itself stored as a Jinja Template <sup>2</sup>. The prompt is a few-shot description of the task itself. It contains several examples of questions and code outputs, as well as a template "hole" question to fill in with the user question before generating. (This format is inspired by the PromptSource system (Bach et al., 2022))

```
Question:
A robe takes 2 bolts of blue fiber
and half that much white fiber.
How many bolts in total does it take?
Code:
2 + 2/2
Question:
{{question}}
Code:
```

Next we describe the code for running this output in a Python interpreter. We treat all external models in the same manner, so the interface to the python interpreter behaves the same as an LLM. Instead of defining the prompt in its own file, we use the option to write the Jinja code into the decorator.

```
@prompt(Python(),
        template="import math\n{{code}}")
def python(model, code):
    "Call Python interpreter"
    code = "\n".join(
        code.strip().split("\n")[1:-1])
    return model(dict(code=code))
```

Given these two prompt construction functions, in the last step we can apply the chaining to produce our output. The API takes a question and then produces an answer by running the two together.

```
def math_demo(query):
    "Chain them together"
    return python(math_prompt(query))
```

However, while this last step looks natural, the semantics are a bit more complex. The MiniChain library uses lazy streams throughout, so this last step does not call the LLMs, but produces a compound Prompt object. This object has access to the entire graph of prompt operations constructed in this chain, roughly analogous to the backprop graph in autodifferentiation libraries.

To compute the final output for a user query we need to instantiate and run the chain.

```
math_demo("""What is the sum of the
           powers of 3 (3^i) that
           are smaller than 100?
           """).run()
```

## 4 Visualizing the LLM Interactions

Since the system uses lazily instantiated chain of prompts with explicit prompt templates, it has full transparency into each step of the prompting process. This design makes it easy to extract and control the intermediate states of the system before and during runtime.

One particular benefit of this transparency is that it facilitates automatic interactive visualization and debugging. MiniChain includes a full visualization library built in based on the Gradio visualization library <sup>3</sup>. The visualization does not require any additional code, beyond what was shown in the previous section, and can be launched with the following command.

```
show(math_demo,
      examples=[...],
      subprompts=[math_prompt, python],
      out_type="markdown")
```

This command creates an automatic interactive visualization UI, which is shown in Figure 1. Starting from a text input, it kicks off and runs each step of the lazy chain showing the intermediate steps and output from the system.

The visualization shows each of the prompts, responses, model uses and the chain structure. Expanding the '...' will show additional low level information like the raw template, variables used, and the commands for calling the underlying LLM model.

The visualization mode of the library also supports additional modalities. *Minichain* supports the ability to utilize models that are non-text based, such as images, video, and audio. In Figure 2, we have the model first write a story and then chain that with a Stable Diffusion model that draws the output into an image.

In addition to being lazy, the graph used in MiniChain is by default streaming. This means that the visualization can display partial outputs from a call to an LLM in real-time. For a slow model like GPT-4, showing intermediate results as

<sup>2</sup><https://jinja.palletsprojects.com/en/3.0.x/>

<sup>3</sup><https://www.gradio.app/>



Figure 1: Automatic interactive visualization of the Math prompt showing input and outputs.

they are generating presents a much improved user experience. Practically this is configured through a customization of the prompt call. For example, in one of the examples we extract a table from a document as a CSV file. This code will convert the output to a well-formatted HTML table in real-time.

```
def to_html(out):
    return "..."
```

```
@prompt(OpenAI(),
        template_file="table.pmpmt.txt",
        gradio_conf=GradioConf(
            block_output=gr.HTML,
            postprocess_output = to_html)
    )
def extract(model, passage):
    return model(dict(passage=passage))
```

## 5 Use-Cases

*MiniChain* is an opinionated library, and one of the goals is to not build additional features that are not related to chaining into the library. We argue for this minimality, by showing how popular prompt paradigms can be implemented without custom

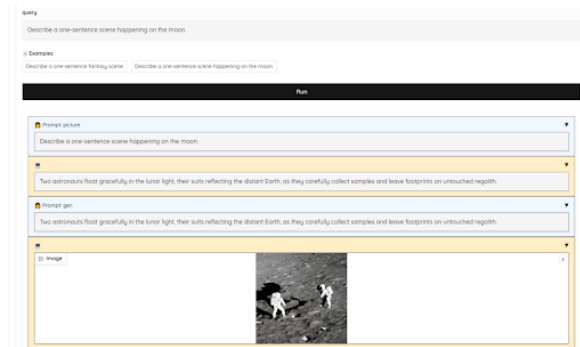


Figure 2: Visualization of a tool-use prompt-chain with image generation.

support.

### 5.1 Retrieval Augmentation

The process of chaining becomes more interesting if we want to allow intermediate processing and computation in addition to prompted calls to large language models. One popular use-case is to store dense embeddings in a vector database in order to support retrieval augmented question answering using large language models. In this section we consider the example of a question answering about the Olympics based <sup>4</sup>. The chain will a) compute the embedding of a passage, b) use it to lookup a corresponding Wikipedia article, c) use this article to answer the question.

We begin with a similar prompt as before, using an embedding template to process the question into a vector embedding with an LLM.

```
@prompt(HuggingFaceEmbed(),
        prompt="emb.pmpmt.tpl")
def embed(model, query):
    return model(dict(question=query))
```

Next we need to look up this embedding in our vector database. Oftentimes it is overkill to have a full vector database, so we can simply use a local in-memory matrix to do this lookup. Here we use HuggingFace Datasets (Lhoest et al., 2021) to create our “vector database” by adding a FAISS index (Johnson et al., 2017).

```
d = datasets.load_from_disk("oly.data")
d.add_faiss_index("embeddings")
```

To use this vector lookup in the chain, we need to inject Python code into our LLM chain. We do this

<sup>4</sup>Used as an example in [https://github.com/openai/openai-cookbook/blob/main/examples/fine-tuned\\_qa/olympics-2-create-qa.ipynb](https://github.com/openai/openai-cookbook/blob/main/examples/fine-tuned_qa/olympics-2-create-qa.ipynb)

with the `transform()` annotation, which allows us to lift a non-prompt to a function in the LLM chain. This function returns the `k` nearest neighbors.

```
@transform()
def neighbors(inp, k):
    return d.get_nearest_examples(
        "embeddings", np.array(inp), k)
```

Finally we introduce a prompt that uses the nearest neighbors and the original question to answer and construct the chain itself.

```
@prompt(OpenAI(),
        template_file="qa.pmppt.tpl")
def answer(model, query, neighbors):
    return model(dict(question=query,
                      docs=neighbors))

def qa(query):
    n = neighbors(embed(query), 3)
    return answer(query, n)
```

## 5.2 Agents and Tool Use

There has been significant excitement over the development of Agent based systems for LLMs that have the ability to process a confirmation and make use of various tools, such as AutoGPT (AutoGPT). While MiniChain does not have agent based behavior built in, it does have API features that make these systems possible to construct. A “tool” in MiniChain is just represented as having multiple models that can be called in a prompt. Prompts can be setup so that they can dynamically select which tool they should call next. This enables transparency in the visualization, while also maintaining each of use.

More tangibly, if we have a set of tools from some repository such as TaskMatrix (Liang et al., 2023), then we can have the model decide using plain python which to utilize.

```
tools = {tool1, tool2, ... }
@prompt(tools)
def tool_use(model, selector, command):
    return model(command,
                  tool_num=tools[selector])
```

To build an Agent-based system all that is required is to combine this with a prompt and parsing command to determine which tool should be used. Here’s an example of the prompt instructions given to the model and a parsing function.

Thought: Do I need to use a tool? Yes  
Action: the action to take, should be one of

```
[{% for tool in tools.keys()%}
  {{tool[0]}}},
{% endfor %}]
```

Action Input: the input to the action

```
@transform()
def tool_parse(out):
    lines = out.split("\n")
    if lines[0].endswith("Yes"):
        return lines[1], line[2]
    else:
        return Break()
```

Where `Break()` is a command to stop the chain from processing.

## 5.3 Chatbots and Memory

Given the fixed-length memory of LLMs, it is important to utilize available model context for higher-level tasks such as chat-like behavior. As such, libraries for chaining have devoted significant overhead to abstractions of memory to maintain previous contextual information.

*MiniChain* does not directly handle this problem, and instead relies on Python. Since chains are lazy and immutable, they do not provide any mechanism to maintain explicit state. To simulate mutability, the user needs to update and maintain their own history. Python support for easy immutable containers makes this relatively straightforward.

As an example, let us consider a chat example with a model that needs to remember the last `N` responses it has made to the user. We store this in a state data structure.

```
@dataclass(frozen=True)
class State:
    memory: List[Tuple[str, str]]
    human_input: str = ""
    # ...
```

We can then use this memory with a chain where at each step we update the state.

```
@prompt(OpenAI(), template_file="...")
def chat_response(model, state):
    return model(state)

@transform()
```

```
def update(state, outp):
    return state.push(outp.split()[-1])
```

To construct the chain the main loop is just a for-loop passing the new state back to the next iteration of the `chat_response`.

## 6 Experimental Features

### 6.1 Auto-Prompts from Types

Utilizing LLMs in code requires having some certainty as to the intermediate types of the variables being produced. In practice even powerful models like GPT-4 have trouble consistently producing outputs of the expected form. *MiniChain* implements methods for notating and describing types. Specifically it includes a `type_to_prompt` method that allows users to specify specific types that they want the system to extract. It then uses this type specification to describe to the model the format.

```
@dataclass
class Player:
    player: str
    stats: List[Stat]

@prompt(OpenAI(), template_file="...")
def stats(model, passage):
    return model(dict(passage=passage,
                      typ=type_to_prompt(Player)))

@transform()
def to_data(s:str):
    return [Player(**j)
            for j in json.loads(s)]
```

A similar approach was recently implemented in *TypeChat* (Microsoft, b), a system that uses Javascript type annotations to automatically produce prompts and ensure adherence.

### 6.2 Back-tracking

LLMs in code is inherently a non-deterministic process. Even at temperature 0, many LLMs do not return deterministic results<sup>5</sup>. This behavior increases the importance of error checking within the chain itself. The lazy nature of *Minichain* makes it feasible to support handling errors through an explicit failure mechanism, and even allow the chain to back-up and retry its search again. Previous

<sup>5</sup><https://twitter.com/BorisMPower/status/1608522707372740609>

nodes in the chain will be able check the cause of the future failure and even update their prompt. This mechanism can be used to implement error correction by pointing out the failures of the output and revising based on failed output.

## 7 Conclusion

We describe *MiniChain*, a software toolkit for prompt chaining. The library focuses on creating an explicit chain of prompts each of which are simple Python functions. Each prompt separates out the descriptive language from the actual chain logic, and transformation logic can be specified with standard Python code. The system is transparent, which allows automatic interactive visualization with different modalities, streaming, and detailed debugging. The core language is minimal, but powerful enough to implement core prompt paradigms, such as retrieval, chat-bots, and tool-use. We hope this work demonstrates some of the possibilities of prompt programming, and encourages others to think about the APIs of these systems and how LLMs will integrate into software.

## References

- AutoGPT. Auto-GPT: An experimental open-source attempt to make GPT-4 fully autonomous.
- Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesh 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, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M Rush. 2022. *PromptSource: An integrated development environment and repository for natural language prompts*.
- Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting is programming: A query language for large language models. *Proc. ACM Program. Lang.*, 7(PLDI):1946–1969.
- 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*.



- Harrison Chase. 2022. LangChain.
- Dust. dust: Design and deploy large language model apps.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#).
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. [Billion-scale similarity search with GPUs](#).
- Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. [Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP](#).
- 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 M Rush, and Thomas Wolf. 2021. [Datasets: A community library for natural language processing](#).
- Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, Yun Wang, Linjun Shou, Ming Gong, and Nan Duan. 2023. [TaskMatrix.AI: Completing tasks by connecting foundation models with millions of APIs](#).
- Jerry Liu. 2022. LlamaIndex.
- Microsoft. a. semantic-kernel: Integrate cutting-edge LLM technology quickly and easily into your apps.
- Microsoft. b. TypeChat: TypeChat is a library that makes it easy to build natural language interfaces using types.
- Yohei Nakajima. babyagi.
- Ankit Pal. 2022. Promptify: Structured output from LLMs. <https://github.com/prompts-lab/Promptify>.
- Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. [SCROLLS: Standardized Comparison over long language sequences](#).
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-GPT: Solving AI tasks with ChatGPT and its friends in hugging face](#).

# Okapi: Instruction-tuned Large Language Models in Multiple Languages with Reinforcement Learning from Human Feedback

Viet Dac Lai<sup>1</sup>, Chien Van Nguyen<sup>1</sup>, Nghia Trung Ngo<sup>1</sup>, Thuat Nguyen<sup>1</sup>

Franck Dernoncourt<sup>2</sup>, Ryan A. Rossi<sup>2</sup>, Thien Huu Nguyen<sup>1</sup>

<sup>1</sup>Dept. of Computer Science, University of Oregon, OR, USA

<sup>2</sup>Adobe Research, USA

{vietl@cs,chienn,nghian@cs,thien@cs}@uoregon.edu

{franck.dernoncourt,ryrossi}@adobe.com

## Abstract

A key technology for large language models (LLMs) involves instruction tuning that helps align the models' responses with human expectations to realize impressive learning abilities. Two major approaches for instruction tuning characterize supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF), which are applied to produce the best commercial LLMs. To improve the accessibility of LLMs, various instruction-tuned open-source LLMs have also been introduced recently. However, existing open-source LLMs have only been instruction-tuned for English and a few popular languages, thus hindering their accessibility to many other languages in the world. In addition, SFT has been used as the only approach to instruction-tune open-source LLMs for multiple languages. This has left a significant gap for fine-tuned LLMs based on RLHF in diverse languages and raised important questions on how RLHF can boost the performance of multilingual instruction tuning. To overcome this issue, we present Okapi, the first system with instruction-tuned LLMs based on RLHF for multiple languages. Okapi introduces instruction and response-ranked data in 26 diverse languages to facilitate the experiments and development of future multilingual LLM research. We also present benchmark datasets to enable the evaluation of generative LLMs in multiple languages. Our experiments demonstrate the advantages of RLHF for multilingual instruction over SFT for different base models and datasets. Our framework with created resources, fine-tuned LLMs, interaction scripts are released at <https://github.com/nlp-uoregon/Okapi>. A demo video to show our framework can also be found at: <https://youtu.be/QFV2fkPwvi0>.

## 1 Introduction

Pre-trained on massive data, large language models (LLMs) with hundreds of billions of parameters such as GPT-3 (Rae et al., 2021) can unlock

new emergent abilities that cannot be achieved with smaller models (Wei et al., 2022; Choi et al., 2023; Jiao et al., 2023). However, as LLMs are trained with the autoregressive learning objective, they might exhibit unintended behaviours from human expectations (Tamkin et al., 2021; Weidinger et al., 2021; Kenton et al., 2021). To overcome this issue, instruction fine-tuning has been proposed as a prominent approach to improve capabilities in following human instructions for LLMs and align them with human intentions in conversations (Christiano et al., 2017; Stiennon et al., 2020; Sanh et al., 2021; Ouyang et al., 2022). As such, two major techniques for instruction tuning feature supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) that are leveraged by the best commercial LLMs such as ChatGPT and GPT-4 to deliver outstanding dialog performance.

Another issue with LLMs pertains to the massive scales and closed-source nature of the commercial LLMs that greatly restrict accessibility and the extent of interactions with the technology. To this end, there have been growing efforts from the open-source community to create more accessible LLMs with affordable scales while securing competitive performance as the proprietary LLMs, e.g., LLaMA (Touvron et al., 2023), StableLM (StabilityAI, 2023), Falcon (Almazrouei et al., 2023), and MTP (MosaicML, 2023). Instruction tuning has also been applied to these open-source LLMs to improve their abilities to engage with human, and different instruction datasets have been collected to facilitate the process, e.g., Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), LaMini-LM (Wu et al., 2023), and Dolly (Conover et al., 2023).

However, the instruction-following abilities of existing open-source LLMs have been developed mainly for English and some popular languages (i.e., using instruction data for those languages), failing to support many other languages of the world to serve a broader population (Taori et al.,

2023; Wu et al., 2023). To overcome this challenge, a few contemporary frameworks have explored instruction tuning of LLMs for multiple languages, i.e., Phoenix (Chen et al., 2023) and Bactrian-X (Li et al., 2023). However, their multilingual instruction tuning efforts are limited to only supervised fine-tuning, which is unable to examine reinforcement learning with human feedback (RLHF) to further boost the performance for multilingual LLMs.

To fill in this gap, our work aims to develop Okapi, an open-source framework with RLHF-based instruction-tuned LLMs for multiple languages to provide resources and shed light on their performance for multilingual LLM learning. Okapi will emphasize on less studied languages and open-source LLMs to better democratize the benefits of instruction-tuned LLMs. In particular, an example in the instruction datasets involves an instruction, an input text, and a desired response output/demonstration. In SFT, the pre-trained LLMs are fine-tuned over the instruction triples (*instruction, input, output*) via supervised learning to promote their alignment with human expectations. In RLHF, generated outputs from the SFT-tuned LLMs are first ranked to provide training signals for the reward functions. Afterward, the SFT-tuned models will be further optimized via reinforcement learning utilizing rewards from the trained reward models. As such, RLHF has been successfully employed to create effective commercial LLMs (e.g., InstructGPT, ChatGPT), owing to its ability to learn beyond positive examples associated with only desired demonstrations. By leveraging the reward models, RLHF can observe lower ranking scores for less accurate demonstrations to obtain richer training signals for LLMs. To our knowledge, Okapi is the first work to perform instruction tuning with RLHF for open-source LLMs over multiple languages.

To develop Okapi, we need to overcome the scarcity of instruction datasets in multiple languages to train and evaluate RLHF models. Motivated by the 52K instructions from Alpaca (Taori et al., 2023), we leverage Self-Instruct (Wang et al., 2023) to generate 106K additional instructions in English, introducing a larger dataset to facilitate RLHF evaluation. Afterward, we utilize ChatGPT to translate the instructions into a diverse set of 26 languages, including high-, medium-, and low-resource languages (e.g., Telugu, Ukrainian, Nepali, and Kannada) to offer comprehensive re-

sources and insights for multilingual instruction-tuning. In addition, we introduce a translation-based prompt for ChatGPT to produce rankings for multiple responses of the same instructions from the LLMs, which will be used to train the reward models for RLHF experiments. Finally, we obtain the multilingual evaluation datasets for our fine-tuned LLMs by translating three benchmark datasets for LLMs in the widely-used HuggingFace Open LLM Leaderboard (HuggingFace, 2023; Gao et al., 2021) into 26 languages, i.e., ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and MMLU (Hendrycks et al., 2021).

Using BLOOM (Scao et al., 2022) and LLaMa (Touvron et al., 2023) as the base LLMs, our experiments illustrate that RLHF generally performs better than SFT for multilingual instruction tuning. We also highlight the greater challenges of low-resource languages for multilingual instruction-tuning of LLMs that can be focused in future research. Finally, we release our framework with the created resources and fine-tuned RLHF models. We also provide scripts to interact with our models at <https://github.com/nlp-uoregon/Okapi>.

## 2 Data Preparation

A key requirement for our development of instruction-tuned LLMs with RLHF involves instruction, ranking, and evaluation datasets in multiple languages. To this end, we perform a comprehensive data collection process to prepare necessary data for our multilingual framework Okapi in 26 languages, divided into four major steps: English instruction generation, instruction translation, ranking data production, and evaluation data creation.

**English Instruction Generation:** An instruction example to tune LLMs often has three components: an instruction to specify the task, an input text, and an associated output text (i.e., demonstration or label) (Ouyang et al., 2022). As such, current public instruction datasets for LLMs mainly cover English or some popular languages. Also, we note that a few recent instruction datasets such as xP3 (Muennighoff et al., 2022) and Flan (Chung et al., 2022; Longpre et al., 2023) include multilingual data; however, their instructions are still written in English. Additionally, these datasets tend to be converted from NLP datasets with template instructions, which cannot reflect the flexibility of human-written prompts (Wang et al., 2023). Consequently, our goal is to develop instruction datasets

with instructions, inputs, and output texts in multiple languages, including low-resource ones, to better realize general prompts from human.

To achieve this goal, our strategy is to first obtain English instructions and then translate them into other languages. The benefits of our approach involve consistent instruction content across languages to facilitate performance comparison while taking advantages of translation systems to enable examination for more languages. As such, to conveniently scale our data, we follow the instruction generation method in Alpaca, which in turn employs the Self-Instruct procedure in (Wang et al., 2023), to produce our English dataset.

Starting with a pool of 175 human-written seed instructions in English, at each time, Alpaca samples several instructions from the seeds to form an in-context example to prompt the text-davinci-003 model of OpenAI for new instruction generation. Overall, Alpaca releases 52K instructions for tuning LLMs. In this work, we apply the same Self-Instruct procedure as Alpaca to generate 106K additional English instructions, resulting in a larger combined dataset of 158K instructions for our RLHF-based models in Okapi. Notably, we condition our generation process on the 52K instructions from Alpaca so a new instruction is only saved if it is different enough from Alpaca’s and previous instructions per the ROUGE score criteria in Alpaca (Taori et al., 2023).

**Instruction Translation:** Given the 158K English instructions, we aim to translate them into multiple other languages to obtain data for our multilingual models in Okapi. Table 1 presents 26 selected languages in our framework. Using the data ratios  $r$  of the languages in CommonCrawl<sup>1</sup> to classify languages as in previous work (Bang et al., 2023; Lai et al., 2023), our study encompasses a diverse set of languages, including 8 high-resource languages ( $r > 1.0$ ), 11 medium-resource languages ( $r > 0.1$ ), and 7 low-resource languages ( $r < 0.1$ ). Notably, several of our languages, such as Marathi, Gujarati, and Kannada, have received limited attention in NLP and instruction-tuning.

We utilize ChatGPT to translate the 158K English instructions into 26 target languages for Okapi. Compared to traditional machine translation systems, an advantage of ChatGPT is the ability to use prompts to specify different expectations for the translated texts to facilitate diverse

| Language   | Code | Pop.<br>(M) | CC Size |      | B | L |
|------------|------|-------------|---------|------|---|---|
|            |      |             | (%)     | Cat. |   |   |
| English    | en   | 1,452       | 45.8786 | H    | ✓ | ✓ |
| Russian    | ru   | 258         | 5.9692  | H    | ✓ | ✓ |
| German     | de   | 134         | 5.8811  | H    | ✓ | ✓ |
| Chinese    | zh   | 1,118       | 4.8747  | H    | ✓ |   |
| French     | fr   | 274         | 4.7254  | H    | ✓ | ✓ |
| Spanish    | es   | 548         | 4.4690  | H    | ✓ | ✓ |
| Italian    | it   | 68          | 2.5712  | H    | ✓ | ✓ |
| Dutch      | nl   | 30          | 2.0585  | H    | ✓ | ✓ |
| Vietnamese | vi   | 85          | 1.0299  | H    | ✓ |   |
| Indonesian | id   | 199         | 0.7991  | M    | ✓ |   |
| Arabic     | ar   | 274         | 0.6658  | M    | ✓ |   |
| Hungarian  | hu   | 17          | 0.6093  | M    | ✓ | ✓ |
| Romanian   | ro   | 29          | 0.5637  | M    | ✓ | ✓ |
| Danish     | da   | 6           | 0.4301  | M    | ✓ | ✓ |
| Slovak     | sk   | 7           | 0.3777  | M    | ✓ | ✓ |
| Ukrainian  | uk   | 33          | 0.3304  | M    | ✓ | ✓ |
| Catalan    | ca   | 10          | 0.2314  | M    | ✓ | ✓ |
| Serbian    | sr   | 12          | 0.2205  | M    | ✓ | ✓ |
| Croatian   | hr   | 14          | 0.1979  | M    | ✓ | ✓ |
| Hindi      | hi   | 602         | 0.1588  | M    | ✓ |   |
| Bengali    | bn   | 272         | 0.0930  | L    | ✓ |   |
| Tamil      | ta   | 86          | 0.0446  | L    | ✓ |   |
| Nepali     | ne   | 25          | 0.0304  | L    | ✓ |   |
| Malayalam  | ml   | 36          | 0.0222  | L    | ✓ |   |
| Marathi    | mr   | 99          | 0.0213  | L    | ✓ |   |
| Telugu     | te   | 95          | 0.0183  | L    | ✓ |   |
| Kannada    | kn   | 64          | 0.0122  | L    | ✓ |   |

Table 1: List of 26 non-English languages in Okapi along with their codes, numbers of first and second speakers (the “Pop.” column), data ratios in CommonCrawl, and categories. The languages are grouped into categories based on their data ratios in CommonCrawl: High- (H,  $> 1\%$ ), Medium- (M,  $> 0.1\%$ ), and Low-Resource (L,  $> 0.01\%$ ). Columns “B” and “L” indicate if a language is supported by the LLMs BLOOM and LLaMa (respectively) or not.

types of instructions. For example, we can instruct ChatGPT to preserve code in the instruction examples about programming as we expect code to be the same in the instructions across natural languages. It is important to note that we directly translate the instruction, input text, and associated output in each English instruction of our data. This is in contrast to the other multilingual instruction-tuning approaches (Li et al., 2023) that only translate instructions and input texts into a target language (using Google Translate), and then prompt ChatGPT to generate response outputs in the target language based on the translated instructions and inputs. The intuition for our approach concerns various potential issues of ChatGPT, e.g., hallucination, bias, mathematical reasoning, and toxic content (Bang et al., 2023; Borji, 2023), that can be exaggerated if ChatGPT is used to produce responses in non-English languages for different tasks (Lai et al., 2023). By generating the instructions and responses in English, we aim to capitalize on the greater performance of LLMs for different

<sup>1</sup><http://commoncrawl.org>

**Translation Prompt:** Translate the values in the following JSON object into <target language> language. You must keep the keys in the JSON object in English. If a value contains programming code, only translate the comments while preserving the code. Your translations must convey all the content in the original text and cannot involve explanations or other unnecessary information. Please ensure that the translated text is natural for native speakers with correct grammar and proper word choices. Your translation must also use exact terminology to provide accurate information even for the experts in the related fields. Your output must only contain a JSON object with translated text and cannot include explanations or other information.

Figure 1: Translation prompt for ChatGPT for multiple languages in Okapi. We organize our instruction examples into JSON objects with fields for translation prompts, instructions, inputs, and outputs send to ChatGPT. <target language> is replaced with the selected languages in our dataset.

NLP tasks in English to avoid the exaggeration issues and achieve higher quality instructions.

**Ranking Data Production:** To perform RLHF for a LLM, we need to obtain ranked response outputs from the model for the same instruction and input to train a reward model. Concretely, given a LLM  $M$  and a dataset  $S = \{inst_k, input_k\}_{k=1}^N$  with  $N$  pairs of instructions  $inst_k$  and input texts  $input_k$  for a target language, we first prompt  $M$  to generate  $T$  output responses  $output_k = \{output_k^1, \dots, output_k^T\}$  for each pair of instruction and input text  $(inst_k, input_k)$  ( $T > 1$ ). Afterward, the responses in  $output_k$  are ranked according to their fitness and quality for the instruction  $inst_k$  and input text  $input_k$ . This ranking data  $\{inst_k, input_k, output_k\}$  can then be leveraged to train our reward models in Okapi.

We also employ ChatGPT to rank the response outputs for multilingual LLMs. Similar to the motivation for our translation-based approach to obtain instruction data in multiple languages, our ranking strategy first asks ChatGPT to translate the instructions and responses  $\{inst_k, input_k, output_k\}$  of a target language into English; the ranking of the responses is then done over the translated English data to exploit the greater quality of ChatGPT for English (using the translation and ranking prompts in Figure 2). For each example  $\{inst_k, input_k, output_k\}$ , the translation and ranking prompts are wrapped in a two-turn dialog with ChatGPT to allow the ranking process to condition on the resulting translations. It also ensures the same output format for the ranking prompts for convenient parsing. Overall, we obtain ranked response outputs for 42K instructions from the 106K

•**Turn 1: Translation Prompt** You will be given an instruction, an input for the instruction, and four possible responses for the instruction. The input can be empty, shown as <empty>. You need to translate the provided instruction, input, and responses into English.

*Instruction: ...*  
*Input: ...*  
*Response 1: ...*  
*Response 2: ...*  
*Response 3: ...*  
*Response 4: ...*

• **Turn 2: Ranking Prompt** Given the translated instruction, input, and responses, you will need to rank the responses according to three factors: correctness with respect to the instruction and input, coherence, and naturalness.

You will need to provide an overall rank for each response when all the three factors are considered. The overall rank for a response must be an integer between 1 and 4 where 1 is for the best response and 4 is the worst response. You cannot assign the same rank for two different responses.

The format of your output must be: for each response: "<Response  $r$ >: overall rank: <1/2/3/4>". The responses must be in original order. Do not include explanation in your output.

**An Example Output from ChatGPT:**

Response 1: 3  
 Response 2: 1  
 Response 3: 4  
 Response 4: 2

Figure 2: Prompts to translate and rank responses.

generated instructions for each language in Okapi.

**Evaluation Data Creation:** We employ three datasets in the HuggingFace Open LLM Leaderboard (HuggingFace, 2023) i.e., ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and MMLU (Hendrycks et al., 2021), to evaluate the model performance for our Okapi framework. All the datasets are organized as multiple-choice question-answering tasks although they focus on different types of knowledge and reasoning aspects. ARC involves 1170 grade-school science questions; HellaSwag provides 9162 commonsense inference questions that are easy for humans, but difficult for many state-of-the-art models; and MMLU assesses accuracy for 13062 questions over various branches of knowledge (STEM, humanities, social sciences, and more). Nevertheless, although the LLM community has widely adopted the HuggingFace leaderboard for performance examination, the datasets are only provided for English, thus unable to evaluate LLMs for the languages in our work. To this end, we translate the examples of the three datasets into 26 selected languages using ChatGPT and the translation prompt in Figure 1. The translated datasets are then reserved to evaluate the LLMs in our Okapi framework.

### 3 Instruction-tuning with RLHF

We follow three steps to develop a fine-tuned LLM with RLHF for each target language in our Okapi framework: supervised fine-tuning, reward model training, and reinforcement learning.

**Supervised Fine-tuning (SFT):** Starting with a multilingual LLM as the base, e.g., BLOOM (Scao et al., 2022), we fine-tune the model with our instruction dataset for the target language using supervised learning with the autoregressive objective. Here, we fine-tune the entire base LLM for all of its parameters with SFT to accurately understand the model performance for multilingual settings.

**Reward Model Training:** The goal of this step is to train a reward model for the target language that will compute reward signals for reinforcement learning to further optimize the SFT-tuned model from the previous step. For each pair of a prompt and potential response, our reward model returns a scalar value to quantify the appropriateness of the response with respect to the instruction and input text in the prompt. We exploit the instructions with multiple ranked responses in the data collection step for this training step. An example to train our reward model for a language involves an instruction and an input text (to form a prompt  $x$ ) along with two sampled responses  $y_c$  and  $y_r$  for  $x$  from our datasets. Based on the ranking information, we can assume one of the responses (i.e.,  $y_c$ ) is more preferable than the other (i.e.,  $y_r$ ). In the next step, the binary ranking loss (Ouyang et al., 2022) is employed to train our reward model, aiming to assign a higher score  $r(x, y_c)$  for the preferred response  $y_c$  than the score  $r(x, y_r)$  for  $y_r$ :  $L_{reward}(\theta) = -\mathbb{E}_{(x, y_c, y_r)} [\log \sigma(r_\theta(x, y_c) - r_\theta(x, y_r))]$ .

**Reinforcement Learning (RL):** With the reward model established for the target language, the SFT model undergoes additional fine-tuning through RL to align it with human preferences. For this purpose, we employ the Proximal Policy Optimization (PPO) algorithm (Ouyang et al., 2022) that maximizes the mean reward of the model via the objective:  $L_{RL}(\phi) = -\mathbb{E}_{x \sim D_{RL}, y \sim \pi_\phi(y|x)} [r_\theta(x, y) - \beta KL(x, y)]$ . Here,  $D_{RL}$  corresponds to the prompt distribution, and  $\pi_\phi(y|x)$  denotes the policy or language model that requires optimization.  $\pi_\phi(y|x)$  is initialized with the SFT-tuned model  $\pi_\phi(y|x)$ . Also,  $KL(x, y) = D_{KL}(\pi_\phi(y|x) || \pi_0(y|x))$  is the Kullback–Leibler divergence to penalize large deviation of  $\pi_\phi$  from the initial SFT policy  $\pi_0$ .

### 4 Experiments

Our Okapi framework utilizes two multilingual LLMs: BLOOM (Scao et al., 2022) and LLaMA (Touvron et al., 2023) as the base models for the fine-tuning processes. We focus on their 7B-parameter versions to facilitate the computing resources and achieve fairer comparison. For each base model and target language, we carry out both SFT-based and RLHF-based instruction-tuning:

- SFT: The base model is fine-tuned over our entire set of 158K translated instructions for the target language in the supervised manner.

- RLHF: The base model is first fine-tuned with supervised training over 52K translated instructions from Alpaca. Afterward, a reward model is trained using the 42K instructions with ranked responses obtained in the data collection. Note that the ranked responses are sampled from the SFT-tuned base model over the 52K Alpaca instructions from previous step. Finally, given the reward model, the SFT-tuned model is further optimized via reinforcement learning over the 64K remaining translated instructions from our generation set.

Following the HuggingFace Open LLM Leaderboard, the Eleuther AI Language Model Evaluation Harness framework (Gao et al., 2021) is used to compute the model performance over the translated datasets ARC, HellaSwag, and MMLU for each language in our framework. As a reference, we also report the performance of the base models BLOOM and LLaMA in the experiments. Finally, for BLOOM, we further compare with BLOOMZ (Muennighoff et al., 2022), which is the fine-tuned version of BLOOM over the cross-lingual task mixture dataset xP3 with millions of multilingual instructions to achieve instruction-following ability.

**Evaluation:** Tables 2 and 3 present the performance of the models on ARC, HellaSwag, and MMLU when BLOOM and LLaMa are used as the base models (respectively). In the tables, for each language group (i.e., high-, medium-, and low-resource), we report the average performance over the languages and the performance for two example languages in the group. We also include the average performance over all languages in Okapi. As some of our languages in Okapi (especially the low-resource ones) are not supported by LLaMA, Table 3 will omit those languages (see Table 1). Finally, Appendix A provides performance of the models over all languages and datasets in Okapi.

The first observation from the tables is that

| Data        | Language       | BLOOM       | BLOOMZ | SFT         | RLHF        |
|-------------|----------------|-------------|--------|-------------|-------------|
| ARC         | Chinese        | 37.3        | 37.0   | 37.9        | <b>40.0</b> |
|             | French         | 36.7        | 37.6   | 37.6        | <b>41.2</b> |
|             | Average High   | 31.5        | 30.7   | 32.3        | <b>34.0</b> |
|             | Indonesian     | 36.0        | 35.9   | 37.4        | <b>38.8</b> |
|             | Arabic         | 31.4        | 31.2   | 32.1        | <b>33.2</b> |
|             | Average Medium | 27.7        | 26.7   | 28.0        | <b>29.8</b> |
|             | Bengali        | 26.2        | 25.5   | 26.8        | <b>28.9</b> |
|             | Kannada        | <b>24.7</b> | 24.6   | 24.5        | 24.6        |
|             | Average Low    | 25.1        | 24.9   | 24.7        | <b>25.6</b> |
| Average All | 28.2           | 27.4        | 28.4   | <b>30.0</b> |             |
| HellaSwag   | Chinese        | 51.2        | 42.6   | 51.8        | <b>53.8</b> |
|             | French         | 56.6        | 45.7   | 55.9        | <b>58.7</b> |
|             | Average High   | 43.8        | 39.6   | 44.5        | <b>46.6</b> |
|             | Indonesian     | 49.5        | 42.0   | 50.0        | <b>52.2</b> |
|             | Arabic         | 43.3        | 39.5   | 44.3        | <b>47.0</b> |
|             | Average Medium | 35.7        | 33.5   | 36.9        | <b>38.9</b> |
|             | Bengali        | 32.8        | 31.5   | 33.9        | <b>35.4</b> |
|             | Kannada        | 30.3        | 30.9   | 30.7        | <b>32.1</b> |
|             | Average Low    | 30.3        | 30.9   | 31.2        | <b>32.3</b> |
| Average All | 36.8           | 34.7        | 37.7   | <b>39.5</b> |             |
| MMLU        | Chinese        | <b>29.1</b> | 27.2   | 27.7        | 28.2        |
|             | French         | 27.4        | 27.7   | 27.7        | <b>28.4</b> |
|             | Average High   | <b>27.5</b> | 26.4   | 26.9        | <b>27.5</b> |
|             | Indonesian     | 26.9        | 26.3   | 26.8        | <b>27.5</b> |
|             | Arabic         | 27.5        | 24.4   | 27.4        | <b>27.7</b> |
|             | Average Medium | <b>27.1</b> | 25.8   | 26.7        | <b>27.1</b> |
|             | Bengali        | <b>28.2</b> | 25.9   | 27.1        | 26.8        |
|             | Kannada        | 26.7        | 26.0   | 26.6        | <b>26.8</b> |
|             | Average Low    | <b>26.7</b> | 25.9   | 26.1        | 26.1        |
| Average All | <b>27.1</b>    | 26.0        | 26.6   | 26.9        |             |

Table 2: Performance of the models using BLOOM 7B.

| Data        | Language       | LLaMA | SFT         | RLHF        |
|-------------|----------------|-------|-------------|-------------|
| ARC         | German         | 35.1  | 37.5        | <b>39.7</b> |
|             | French         | 37.3  | 38.4        | <b>38.8</b> |
|             | Average High   | 35.1  | 36.5        | <b>38.7</b> |
|             | Danish         | 32.7  | 35.1        | <b>36.8</b> |
|             | Ukrainian      | 32.9  | 35.7        | <b>36.4</b> |
|             | Average Medium | 32.0  | 34.3        | <b>36.2</b> |
| Average All | 33.3           | 35.2  | <b>37.3</b> |             |
| HellaSwag   | German         | 49.9  | 49.0        | <b>52.6</b> |
|             | French         | 55.7  | 55.6        | <b>56.9</b> |
|             | Average High   | 51.4  | 51.2        | <b>53.7</b> |
|             | Danish         | 46.7  | 47.7        | <b>51.7</b> |
|             | Ukrainian      | 44.1  | 46.9        | <b>47.7</b> |
|             | Average Medium | 42.7  | 44.0        | <b>46.5</b> |
| Average All | 46.4           | 47.1  | <b>49.6</b> |             |
| MMLU        | German         | 29.9  | 30.4        | <b>31.7</b> |
|             | French         | 30.5  | <b>31.0</b> | 30.7        |
|             | Average High   | 30.1  | 30.4        | <b>30.9</b> |
|             | Danish         | 30.0  | 30.9        | <b>31.8</b> |
|             | Ukrainian      | 29.4  | 30.8        | <b>31.6</b> |
|             | Average Medium | 29.5  | 29.9        | <b>30.7</b> |
| Average All | 29.8           | 30.1  | <b>30.8</b> |             |

Table 3: Performance of the models using LLaMa 7B.

RLHF is generally better than SFT for multilingual fine-tuning of LLMs over different datasets, base models, and language groups. It is also evident that the RLHF-tuned models can significantly improve the performance of the original base models (i.e., BLOOM and LLaMa) for almost all the language groups and datasets. In all, it highlights the quality of the generated instruction data and the effectiveness of RLHF in Okapi.

Comparing the performance across language groups, the models tend to achieve the highest performance for the high-resource languages, followed by the medium-resource and low-resource

languages. The performance improvement of RLHF for low-resource languages is also the least (based on BLOOM). Interestingly, our fine-tuned BLOOM models with 158K generated instructions can significantly outperform BLOOMZ over almost all the languages for the ARC, HellaSwag, and MMLU datasets using either SFT or RLHF. As BLOOMZ has fine-tuned BLOOM over more than 78M multilingual instructions converted from NLP datasets (Muennighoff et al., 2022), it demonstrates the higher quality of our generated instructions for multilingual instruction tuning of LLMs.

## 5 Related Work

The most advanced methods for NLP involve fine-tuning the pre-trained language models (PLMs) on training data of the downstream tasks (Min et al., 2023). Instruction tuning can be considered as a special type of fine-tuning techniques for PLMs where generative PLMs (e.g., GPT) are further trained with instruction data to accomplish the instruction following abilities. SFT is the most popular instruction tuning approach that is leveraged by most of the existing LLMs, including ChatGPT, Apaca (Taori et al., 2023), and Vicuna (Chiang et al., 2023). RLHF can also be used to further enhance LLMs (Wei et al., 2021; Ouyang et al., 2022) although it has been less explored by current open-source LLMs due to the challenges in obtaining ranking data for the reward models. For multilingual learning, instruction tuning is only applied in the form of SFT for non-English languages using multilingual LLMs, e.g., BLOOM and LLaMa, in a few contemporary work (Chen et al., 2023; Li et al., 2023; Muennighoff et al., 2022).

## 6 Conclusion

We present the first framework, called Okapi, on instruction tuning for LLMs in multiple language using RLHF. We introduce instruction, ranked response, and evaluation data in 26 diverse languages to enable the training of RLHF methods. Our results reveal the benefits of RLHF for multilingual fine-tuning of LLMs and the challenging problems of low-resource languages in this area.

## Acknowledgement

This research has been supported by the Army Research Office (ARO) grant W911NF-21-1-0112, the NSF grant CNS-1747798 to the IUCRC Center for Big Learning, and the NSF grant # 2239570.

## Ethical Statement

Our framework utilizes the multilingual LLMs BLOOM-7B and LLaMa-7B to develop instruction-tuned models with reinforcement learning from human feedback. To obtain necessary resources to train and evaluate our models, we also apply Self-Instruct (Taori et al., 2023) with GPT-3 to generate English instruction data, and ChatGPT to translate and rank our response data in different languages. As such, the models in our framework might inherit potential issues in the underlying models of BLOOM, LLaMa, GPT-3, and ChatGPT, such as hallucination, biases, and toxic content. Regrettably, the data required to train such LLMs, even in the case of purportedly open-source models such as LLaMa and BLOOM, remains unreleased to enable essential investigation into these matters for our models. Future research can explore open-source datasets, such as CulturaX (Nguyen et al., 2023) and RedPajama (Computer, 2023), to develop truly open LLMs, enabling deeper attribution of the problems and better understanding of the models' operations. To maximally minimize the impacts of these issues in the current work, our framework will fully release the generated instruction, ranking, and evaluation data to enable comprehensive exploration and research for the techniques. We will also restrict the release of our models to research purpose, respecting the policy of the underlying models such as LLaMa and ChatGPT, to facilitate future research for LLMs while limiting the potential ethical issues for the society. Consequently, we do not believe our framework poses any greater societal risks than existing published research in this area for LLMs (Wang et al., 2023). Finally, we confirm that our work fully complies with the ACL Ethics Policy and there is no other ethical issues associated with our work, to the best of our knowledge.

## References

Ebtesam Almazrouei, Hamza Alobeidli, and Abdulaziz Alshamsi et al. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *ArXiv*, abs/2302.04023.

Ali Borji. 2023. A categorical archive of chatgpt failures. *ArXiv*, abs/2302.03494.

Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. Phoenix: Democratizing chatgpt across languages. *ArXiv*, abs/2304.10453.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality.

Jonathan Choi, Kristin Hickman, Amy Monahan, and Daniel Schwarcz. 2023. Chatgpt goes to law school. Available at SSRN.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Hyung Won Chung, Le Hou, S. Longpre, and Barret Zoph et al. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Together Computer. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Mike Conover, Matt Hayes, and Ankit Mathur et al. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. <https://www.databricks.com>.

Leo Gao, Jonathan Tow, Stella Biderman, and et al. 2021. A framework for few-shot language model evaluation.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

HuggingFace. 2023. Open llm leaderboard. [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca).

Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. *ArXiv*, 2301.08745.

Zachary Kenton, Tom Everitt, Laura Weidinger, Iason Gabriel, Vladimir Mikulik, and Geoffrey Irving. 2021. Alignment of language agents. *ArXiv*, abs/2103.14659.



- Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *ArXiv*, abs/2304.05613.
- Haonan Li, Fajri Koto, Minghao Wu, Alham Fikri Aji, and Timothy Baldwin. 2023. Bactrian-x: A multilingual replicable instruction-following model with low-rank adaptation. *ArXiv*, abs/2305.15011.
- S. Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *ArXiv*, abs/2301.13688.
- Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Iana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Survey*.
- MosaicML. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. <https://www.mosaicml.com/blog/mpt-7b>.
- Niklas Muennighoff, Thomas Wang, and Lintang Sutawika et al. 2022. Crosslingual generalization through multitask finetuning. *ArXiv*, abs/2211.01786.
- Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan Rossi, and Thien Huu Nguyen. 2023. Culturax: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. *ArXiv*, abs/2309.09400.
- 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 E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.
- Jack Rae, Sebastian Borgeaud, and et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.
- Victor Sanh, Albert Webson, and Colin Raffel et al. 2021. Multitask prompted training enables zero-shot task generalization. *ArXiv*, abs/2110.08207.
- Teven Scao, Angela Fan, and et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100.
- StabilityAI. 2023. Stablelm: Stability ai language models. <https://github.com/stability-AI/stableLM>.
- Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan J. Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. *ArXiv*, abs/2009.01325.
- Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. 2021. Understanding the capabilities, limitations, and societal impact of large language models. *ArXiv*, abs/2102.02503.
- Rohan Taori, Ishaan Gulrajani, and Tianyi Zhang et al. 2023. Stanford alpaca: An instruction-following llama model. [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca).
- Hugo Touvron, Thibaut Lavril, and Gautier Izacard et al. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. In *Proceedings of the International Conference on Learning Representations*.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.
- Laura Weidinger, John F. J. Mellor, and Maribeth Rauh et al. 2021. Ethical and social risks of harm from language models. *ArXiv*, abs/2112.04359.
- Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2023. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *ArXiv*, abs/2304.14402.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

## A Model Performance

Tables 4, 5, and 6 present the performance of the models on the ARC, HellaSwag, and MMLU datasets (respectively) across all languages when BLOOM is used as the base model. Similarly, Tables 7, 8, and 9 report the performance with the base model LLaMA over the three datasets. In the tables, in addition to the average scores over all languages for the models, we also include the average scores for each group of languages (i.e., rows “Ave Group” for high-, medium-, and low-resource languages) to facilitate the comparisons.

|                  | Language         | BLOOM       | BLOOMZ      | SFT         | RLHF        |
|------------------|------------------|-------------|-------------|-------------|-------------|
| High-Resource    | Russian          | 27.5        | 25.5        | 29.2        | 30.3        |
|                  | German           | 26.3        | 25.4        | 24.9        | 25.5        |
|                  | Chinese          | 37.3        | 37.0        | 37.9        | 40.0        |
|                  | French           | 36.7        | 37.6        | 37.6        | 41.2        |
|                  | Spanish          | 38.1        | 37.2        | 39.7        | 41.5        |
|                  | Italian          | 29.0        | 27.5        | 29.3        | 31.3        |
|                  | Dutch            | 23.1        | 21.5        | 24.8        | 26.1        |
|                  | Vietnamese       | 33.7        | 33.5        | 35.0        | 36.2        |
|                  | <b>Ave Group</b> | <b>31.5</b> | <b>30.7</b> | <b>32.3</b> | <b>34.0</b> |
| Medium-Resource  | Indonesian       | 36.0        | 35.9        | 37.4        | 38.8        |
|                  | Arabic           | 31.4        | 31.2        | 32.1        | 33.2        |
|                  | Hungarian        | 25.9        | 22.8        | 25.2        | 27.5        |
|                  | Romanian         | 26.9        | 23.4        | 27.5        | 30.3        |
|                  | Danish           | 24.6        | 24.6        | 23.6        | 25.2        |
|                  | Slovak           | 24.9        | 22.5        | 26.2        | 27.3        |
|                  | Ukrainian        | 22.8        | 23.1        | 23.6        | 25.2        |
|                  | Catalan          | 34.7        | 35.8        | 35.1        | 38.9        |
|                  | Serbian          | 25.1        | 23.6        | 25.6        | 27.8        |
|                  | Croatian         | 23.7        | 22.8        | 22.7        | 24.1        |
|                  | Hindi            | 29.2        | 28.2        | 28.5        | 29.6        |
| <b>Ave Group</b> | <b>27.7</b>      | <b>26.7</b> | <b>28.0</b> | <b>29.8</b> |             |
| Low-Resource     | Bengali          | 26.2        | 25.5        | 26.8        | 28.9        |
|                  | Tamil            | 24.2        | 25.6        | 23.7        | 25.1        |
|                  | Nepali           | 22.3        | 22.7        | 23.4        | 25.7        |
|                  | Malayalam        | 26.4        | 25.1        | 24.6        | 24.7        |
|                  | Marathi          | 27.3        | 24.8        | 25.8        | 26.0        |
|                  | Telugu           | 24.3        | 25.8        | 23.9        | 24.5        |
|                  | Kannada          | 24.7        | 24.6        | 24.5        | 24.6        |
|                  | <b>Ave Group</b> | <b>25.1</b> | <b>24.9</b> | <b>24.7</b> | <b>25.6</b> |
| <b>Average</b>   | <b>28.2</b>      | <b>27.4</b> | <b>28.4</b> | <b>30.0</b> |             |

Table 4: Performance of the models on the translated ARC dataset over different languages in Okapi. BLOOM 7B is used as the base LLM.

|                  | Language         | BLOOM       | BLOOMZ      | SFT         | RLHF        |
|------------------|------------------|-------------|-------------|-------------|-------------|
| High-Resource    | Russian          | 32.5        | 33.1        | 32.9        | 34.2        |
|                  | German           | 32.4        | 33.1        | 34.7        | 35.9        |
|                  | Chinese          | 51.2        | 42.6        | 51.8        | 53.8        |
|                  | French           | 56.6        | 45.7        | 55.9        | 58.7        |
|                  | Spanish          | 56.7        | 48.7        | 56.1        | 59.0        |
|                  | Italian          | 40.8        | 40.3        | 43.1        | 44.6        |
|                  | Dutch            | 31.7        | 32.3        | 32.6        | 34.9        |
|                  | Vietnamese       | 48.3        | 40.6        | 49.0        | 51.3        |
|                  | <b>Ave Group</b> | <b>43.8</b> | <b>39.6</b> | <b>44.5</b> | <b>46.6</b> |
| Medium-Resource  | Indonesian       | 49.5        | 42.0        | 50.0        | 52.2        |
|                  | Arabic           | 43.3        | 39.5        | 44.3        | 47.0        |
|                  | Hungarian        | 30.1        | 29.8        | 30.8        | 32.7        |
|                  | Romanian         | 31.8        | 32.3        | 33.1        | 35.2        |
|                  | Danish           | 31.2        | 31.5        | 33.8        | 35.7        |
|                  | Slovak           | 29.8        | 29.6        | 31.4        | 32.9        |
|                  | Ukrainian        | 30.0        | 30.4        | 32.2        | 33.6        |
|                  | Catalan          | 51.2        | 40.3        | 50.9        | 53.8        |
|                  | Serbian          | 29.9        | 30.1        | 30.7        | 33.7        |
|                  | Croatian         | 30.0        | 29.4        | 30.5        | 31.6        |
|                  | Hindi            | 36.4        | 34.0        | 37.7        | 39.7        |
|                  | <b>Ave Group</b> | <b>35.7</b> | <b>33.5</b> | <b>36.9</b> | <b>38.9</b> |
|                  | Low-Resource     | Bengali     | 32.8        | 31.5        | 33.9        |
| Tamil            |                  | 29.4        | 29.5        | 30.0        | 30.4        |
| Nepali           |                  | 30.9        | 31.9        | 32.5        | 34.1        |
| Malayalam        |                  | 28.8        | 29.8        | 29.7        | 30.2        |
| Marathi          |                  | 31.0        | 31.9        | 31.7        | 32.5        |
| Telugu           |                  | 29.2        | 30.7        | 30.0        | 31.7        |
| Kannada          |                  | 30.3        | 30.9        | 30.7        | 32.1        |
| <b>Ave Group</b> |                  | <b>30.3</b> | <b>30.9</b> | <b>31.2</b> | <b>32.3</b> |
| <b>Average</b>   | <b>36.8</b>      | <b>34.7</b> | <b>37.7</b> | <b>39.5</b> |             |

Table 5: Performance of the models on the translated HellaSwag dataset over different languages in Okapi. BLOOM 7B is used as the base LLM.

|                 | Language         | BLOOM            | BLOOMZ      | SFT         | RLHF        |             |
|-----------------|------------------|------------------|-------------|-------------|-------------|-------------|
| High-Resource   | Russian          | 26.2             | 25.4        | 26.5        | 26.8        |             |
|                 | German           | 28.1             | 25.6        | 27.0        | 28.6        |             |
|                 | Chinese          | 29.1             | 27.2        | 27.7        | 28.2        |             |
|                 | French           | 27.4             | 27.7        | 27.7        | 28.4        |             |
|                 | Spanish          | 28.9             | 27.1        | 27.8        | 28.1        |             |
|                 | Italian          | 25.7             | 25.8        | 25.1        | 26.0        |             |
|                 | Dutch            | 26.4             | 26.0        | 26.1        | 26.0        |             |
|                 | Vietnamese       | 28.1             | 26.3        | 27.0        | 27.5        |             |
|                 | <b>Ave Group</b> | <b>27.5</b>      | <b>26.4</b> | <b>26.9</b> | <b>27.5</b> |             |
| Medium-Resource | Indonesian       | 26.9             | 26.3        | 26.8        | 27.5        |             |
|                 | Arabic           | 27.5             | 24.4        | 27.4        | 27.7        |             |
|                 | Hungarian        | 26.9             | 26.1        | 25.4        | 26.3        |             |
|                 | Romanian         | 27.4             | 25.9        | 27.6        | 27.4        |             |
|                 | Danish           | 27.1             | 25.2        | 27.2        | 26.9        |             |
|                 | Slovak           | 26.1             | 26.3        | 26.4        | 26.1        |             |
|                 | Ukrainian        | 26.6             | 25.8        | 25.9        | 26.4        |             |
|                 | Catalan          | 28.8             | 26.0        | 26.7        | 27.6        |             |
|                 | Serbian          | 27.2             | 25.7        | 27.5        | 27.6        |             |
|                 | Croatian         | 26.0             | 26.1        | 26.4        | 27.7        |             |
|                 | Hindi            | 27.5             | 25.9        | 26.8        | 26.5        |             |
|                 |                  | <b>Ave Group</b> | <b>27.1</b> | <b>25.8</b> | <b>26.7</b> | <b>27.1</b> |
| Low-Resource    | Bengali          | 28.2             | 25.9        | 27.1        | 26.8        |             |
|                 | Tamil            | 26.6             | 26.7        | 26.1        | 26.0        |             |
|                 | Nepali           | 26.6             | 25.6        | 25.5        | 25.2        |             |
|                 | Malayalam        | 26.4             | 25.2        | 25.8        | 25.8        |             |
|                 | Marathi          | 26.3             | 26.0        | 26.1        | 26.1        |             |
|                 | Telugu           | 26.2             | 25.7        | 25.4        | 25.9        |             |
|                 | Kannada          | 26.7             | 26.0        | 26.6        | 26.8        |             |
|                 |                  | <b>Ave Group</b> | <b>26.7</b> | <b>25.9</b> | <b>26.1</b> | <b>26.1</b> |
|                 |                  | <b>Average</b>   | <b>27.1</b> | <b>26.0</b> | <b>26.6</b> | <b>26.9</b> |

Table 6: Performance of the models on the translated MMLU dataset over different languages in Okapi. BLOOM 7B is used as the base LLM.

|                 | Language       | LLaMA            | SFT         | RLHF        |
|-----------------|----------------|------------------|-------------|-------------|
| High-Resource   | Russian        | 32.1             | 32.8        | 37.7        |
|                 | German         | 35.1             | 37.5        | 39.7        |
|                 | French         | 37.3             | 38.4        | 38.8        |
|                 | Spanish        | 36.8             | 38.7        | 39.3        |
|                 | Italian        | 35.8             | 36.3        | 39.4        |
|                 | Dutch          | 33.6             | 35.2        | 37.5        |
|                 |                | <b>Ave Group</b> | <b>35.1</b> | <b>36.5</b> |
| Medium-Resource | Hungarian      | 29.8             | 31.4        | 33.2        |
|                 | Romanian       | 32.4             | 33.8        | 37.5        |
|                 | Danish         | 32.7             | 35.1        | 36.8        |
|                 | Slovak         | 29.0             | 34.3        | 37.2        |
|                 | Ukrainian      | 32.9             | 35.7        | 36.4        |
|                 | Catalan        | 35.1             | 36.8        | 36.9        |
|                 | Serbian        | 30.8             | 33.5        | 35.8        |
|                 | Croatian       | 33.0             | 33.8        | 35.9        |
|                 |                | <b>Ave Group</b> | <b>32.0</b> | <b>34.3</b> |
|                 | <b>Average</b> | <b>33.3</b>      | <b>35.2</b> | <b>37.3</b> |

Table 7: Performance of the models on the translated ARC dataset over different languages in Okapi. LLaMA 7B is used as the base LLM.

|               | Language        | LLaMA            | SFT         | RLHF        |             |
|---------------|-----------------|------------------|-------------|-------------|-------------|
| High-Resource | Russian         | 45.7             | 46.0        | 49.1        |             |
|               | German          | 49.9             | 49.0        | 52.6        |             |
|               | French          | 55.7             | 55.6        | 56.9        |             |
|               | Spanish         | 56.4             | 55.7        | 56.6        |             |
|               | Italian         | 52.0             | 52.5        | 55.9        |             |
|               | Dutch           | 48.7             | 48.1        | 51.3        |             |
|               |                 | <b>Ave Group</b> | <b>51.4</b> | <b>51.2</b> | <b>53.7</b> |
|               | Medium-Resource | Hungarian        | 37.9        | 38.7        | 41.0        |
| Romanian      |                 | 44.9             | 45.1        | 48.7        |             |
| Danish        |                 | 46.7             | 47.7        | 51.7        |             |
| Slovak        |                 | 35.9             | 39.5        | 43.6        |             |
| Ukrainian     |                 | 44.1             | 46.9        | 47.7        |             |
| Catalan       |                 | 49.6             | 49.2        | 49.0        |             |
| Serbian       |                 | 41.1             | 42.6        | 45.0        |             |
| Croatian      |                 | 41.1             | 42.4        | 45.2        |             |
|               |                 | <b>Ave Group</b> | <b>42.7</b> | <b>44.0</b> | <b>46.5</b> |
|               | <b>Average</b>  | <b>46.4</b>      | <b>47.1</b> | <b>49.6</b> |             |

Table 8: Performance of the models on the translated HellaSwag dataset over different languages in Okapi. LLaMA 7B is used as the base LLM.

|                 | Language         | LLaMA            | SFT         | RLHF        |
|-----------------|------------------|------------------|-------------|-------------|
| High-Resource   | Russian          | 30.2             | 30.0        | 30.6        |
|                 | German           | 29.9             | 30.4        | 31.7        |
|                 | French           | 30.5             | 31.0        | 30.7        |
|                 | Spanish          | 30.3             | 30.4        | 30.9        |
|                 | Italian          | 29.9             | 30.6        | 30.4        |
|                 | Dutch            | 29.8             | 30.0        | 31.1        |
|                 | <b>Ave Group</b> | <b>30.1</b>      | <b>30.4</b> | <b>30.9</b> |
| Medium-Resource | Hungarian        | 29.0             | 29.2        | 30.1        |
|                 | Romanian         | 29.7             | 29.8        | 30.9        |
|                 | Danish           | 30.0             | 30.9        | 31.8        |
|                 | Slovak           | 29.4             | 29.6        | 30.2        |
|                 | Ukrainian        | 29.4             | 30.8        | 31.6        |
|                 | Catalan          | 30.2             | 30.3        | 30.5        |
|                 | Serbian          | 29.2             | 29.7        | 30.4        |
|                 | Croatian         | 29.3             | 29.2        | 30.0        |
|                 |                  | <b>Ave Group</b> | <b>29.5</b> | <b>29.9</b> |
|                 | <b>Average</b>   | <b>29.8</b>      | <b>30.1</b> | <b>30.8</b> |

Table 9: Performance of the models on the translated MMLU dataset over different languages in Okapi. LLaMA 7B is used as the base LLM.

# SAGEViz: Schema GEneration and Visualization

Sugam Devare<sup>\*1</sup>, Mahnaz Koupaee<sup>\*1</sup>, Gautham Gunapati<sup>1</sup>, Sayontan Ghosh<sup>1</sup>, Sai Vallurupalli<sup>2</sup>, Yash Kumar Lal<sup>1</sup>, Francis Ferraro<sup>2</sup>, Nathanael Chambers<sup>3</sup>, Greg Durrett<sup>4</sup>, Raymond Mooney<sup>4</sup>, Katrin Erk<sup>4</sup>, Niranjan Balasubramanian<sup>1</sup>

<sup>1</sup> Stony Brook University, <sup>2</sup> University of Maryland, Baltimore County

<sup>3</sup> United States Naval Academy, <sup>4</sup> The University of Texas at Austin

<sup>1</sup>{sdevare, mkoupaee, ggunapati, sagghosh, ylal, niranjan}@cs.stonybrook.edu

<sup>2</sup>{kolli, ferraro}@umbc.edu, <sup>3</sup>nchamber@usna.edu,

<sup>4</sup>{gdurrett@cs, mooney@cs, katrin.erk@}.utexas.edu

## Abstract

Schema induction involves creating a graph representation depicting how events unfold in a scenario. We present SAGEViz, an intuitive and modular tool that utilizes human-AI collaboration to create and update complex schema graphs efficiently, where multiple annotators (humans and models) can work simultaneously on a schema graph from any domain. The tool consists of two components: (1) a curation component powered by plug-and-play event language models to create and expand event sequences while human annotators validate and enrich the sequences to build complex hierarchical schemas, and (2) an easy-to-use visualization component to visualize schemas at varying levels of hierarchy. Using supervised and few-shot approaches, our event language models can continually predict relevant events starting from a seed event. We conduct a user study and show that users need less effort in terms of interaction steps with SAGEViz to generate schemas of better quality. We also include a video demonstrating the system<sup>1</sup>.

## 1 Introduction

Event schemas are central to understanding and reasoning about events. They provide a way to organize and represent how complex events unfold (Schank and Abelson, 1975; Mooney and DeJong, 1985). Schema-based reasoning enables reliable and explainable prediction of next events, inference of missing events or entities (Chambers and Jurafsky, 2008, 2009; Manshadi et al., 2008; Chambers, 2013; Balasubramanian et al., 2013; Pichotta and Mooney, 2016; Weber et al., 2018; Koupaee et al., 2021; Rezaee et al., 2021), and drawing connections between events that have already occurred (Kwon et al., 2020). For example, when a disease

outbreak happens, it is likely that an investigation into the outbreak and mitigation steps will follow. One main challenge in schema-based reasoning is in acquiring the schematic knowledge at scale.

One approach to automate schema curation is to learn from manually created reference schemas. Manual creation of complex hierarchical schemas require expert annotation, which is time-consuming and not scalable. Further, supervised systems (Ji and Grishman, 2008; Lin et al., 2021) are domain-specific, of poor quality and unable to handle unseen world events<sup>2</sup>.

We propose SAGEViz, a human-in-the-loop schema curation and visualization pipeline, a combined approach to producing human-verified schemas using automated acquisition strategies. We leverage Large Language Models (LLMs) for acquiring various types of events and entity knowledge automatically. We use human inputs to ensure the contextual validity of model produced outputs at various stages in the curation pipeline. SAGEViz’s visualization allows a human curator to easily navigate through the complex event schemas at various levels of granularity starting with the higher level big picture and drilling down to the lower levels or vice versa.

SAGEViz allows users to build and curate a schema from scratch or edit existing ones. SAGEViz begins from a set of domain-expert-identified seed events. For each seed event, an event language model produces a suggested set of *next* events in a systematic fashion. For each participant in the seed event, we generate next events by asking the model to predict the next events in which the participant plays agent-based or patient-based roles (*Event Expansion*).

Human curator selects a subset of valid events

<sup>2</sup>models built till 2019 could not reason about COVID-19 since it had not occurred yet

<sup>\*</sup>Equal contribution

<sup>1</sup><https://github.com/sugampx/SAGEViz>

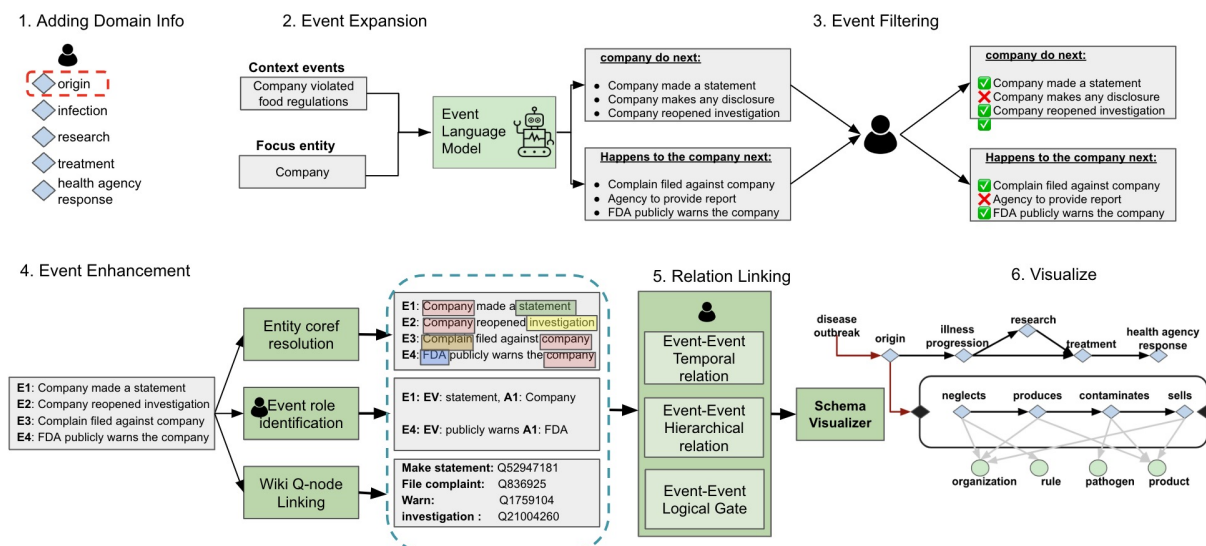


Figure 1: SAGEViz architecture capturing several curation stages and visualization. 1. Expert defines a high-level event structure for the domain 2. Based on some seed event, event language model recursively generates and expands events for the seed. 3. The candidate generated events are filtered by the expert 4. Filtered events are enhanced by adding additional event-specific information 5. Different categories of links are defined between the events 6. The schema at any point can be visualized as a multi-level graph.

from the model generated lists (*Event Filtering*) and enhances it with explanations, event-entity details such as roles, coreference and wikidata links (*Event Enhancement*) before linking event pairs based on logical, temporal and hierarchical event relationships (*Relation Linking*). Human curation in the enhancement stage ensures proper identification of entities with the goal of linking an event to multiple valid related documents. In the linking stage, we identify and validate the relationship between an event pair with the goal of building schemas with a hierarchical structure to support both specific and generic events. SAGEViz’s visualizer displays the schema as a graph at different levels of granularity, showing various event relation types with different widgets and color schemes.

SAGEViz has multiple advantages. First, multiple human annotators can provide input and curate a single complex schema in parallel. Second, it allows for iterative updating, to add, edit and delete non-conforming events at any stage. Third, with SAGEViz allows for modeling complex schemas through quick and intuitive means for generating and expanding sub-events and visualizing relations between them. Last, the back end and front end architecture of the tool allows curators to ensure global coherence of the generated schemas, across iterative updates and multiple simultaneous annotators, leading to more reliable schemas especially

suitable for safety-critical downstream applications.

In summary, our contributions are as follows:

- We present a unified web-based tool to visualize and analyze complex event sequences.
- We leverage various LLMs to create and expand new and existing schemas.
- Our tool enables both human-human and human-AI collaboration for the task of schema generation.

## 2 System Description

In this section we provide a high-level overview and a detailed description of our human-in-the-loop schema curation framework.

### 2.1 Overview

A high-level overview of our system is illustrated in Figure 1. The system has two main components: (i) **Schema curator** - a system for curating the schema through LLM and expert collaboration. (ii) **Schema visualizer** - a system for visualizing the curated schema at any point during curation.

The schema curation process for a complex event starts with a domain expert creating a high-level structure of events to indicate event progression from start to end. For example, for the complex event **pandemic outbreak**, the expert creates a high level structure capturing its

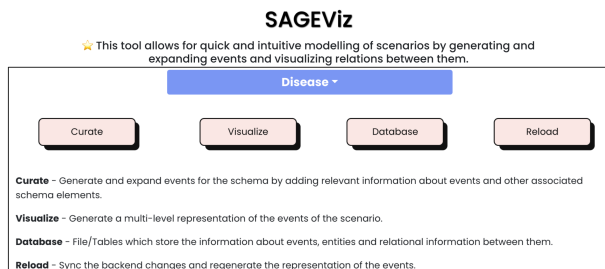


Figure 2: Main interface of SAGEViz. At this stage you can either select an existing domain from the drop-down menu to expand or edit. There are also four actions (along with their descriptions) that can be taken as shown in the figure.

progressive event stages as pathogen\_origin → illness\_progression → investigation → mitigation → treatment → research → legal-actions → health-agency-response. Each of the sub-events is recursively expanded until a desired level of granularity is achieved. While the initial list of sub-events is generated by a domain expert, the candidate subevents for these recursive expansions, are generated using an event language model that is trained for the next event prediction task. The relevant subevents from the candidate subevents are then selected by trained annotators. The selected events are mapped to their canonical form by linking them to corresponding Wikidata qnode entries. For each event, our system also has the provision to record text excerpts from a news article or other online documents, that act as evidence or justification for including the event in the overall schema.

There are multiple types of relationships between events where each relation type uniquely identifies how a pair of complex or sub-events and entities are related. Every event-entity, event-event and entity-entity pairs is related through one of hierarchical (parent, child), temporal (before, after) and logical (OR, AND, XOR) relationships by trained annotators after resolving event coreferences automatically. Our system supports multiple annotators annotating these structures in parallel. The schema visualizer enables the user to view the schema at different hierarchical levels with varying levels of granularity. The various relation types are displayed using different widgets and coloring schemes to easily grasp how the event sequences are related. The annotation process is incremental and iterative where annotators and models can revise an old annotation based on the current state of the overall annotation. The visualizer ensures the annotator can view the current state at all times to

ensure a coherent schema is generated.

## 2.2 Detailed System Description

The following sections will talk about the components of SAGEViz in more detail.

**Event Generator** The first and one of the most important components of the curator tool, is the event generator. These event generators can be finetuned event language models that are trained on the event sequences or can be any other large language model that can generate events (zero-shot or few-shot). The current version of the tool has two event generation models described below, however SAGEViz is designed such that incorporating new models is very easy as long as it uses a similar input/output structure.

*Question-guided event language model:* The first model used in the tool, is a finetuned event language model that takes a set of events and a question about an entity of interest as the context and generates the next event (Koupaee et al., 2023). The users can have control over the entities and can ask the system to generate the events with respect to the desired entities.

*GPT-3.5 one-shot event generator:* Given the efficacy of the large language models such as GPT models (Brown et al., 2020) for generation in zero/few shot settings, we incorporated GPT-3.5 as the second model in the tool.

**Entity Coreference Resolution** Since the questions or prompts to the event generator are entity-based, it is important to track entities across different events. Event generators might generate different mentions of the entities for different events (for example “they” and “attackers” referring to the same entity), however, we assign the same identifier to the various mentions across all events (this allows us to easily track all the events in which the entity “attackers” is involved in).

We incorporate the current state-of-the-art coreference resolution model SpanBERT (Joshi et al., 2019) which identifies clusters of coreferent entity mentions. We assign a unique identifier to all the entities in a cluster and use the identifier when referring to entities in the event sequence leading to more generalized schemas. Since the coreference model does not identify entities with a single mention, we identify noun phrases in the event sequence using the spaCy (Honnibal and Montani, 2017) library and assign unique identifiers to the entities that do not belong to a coreference cluster.

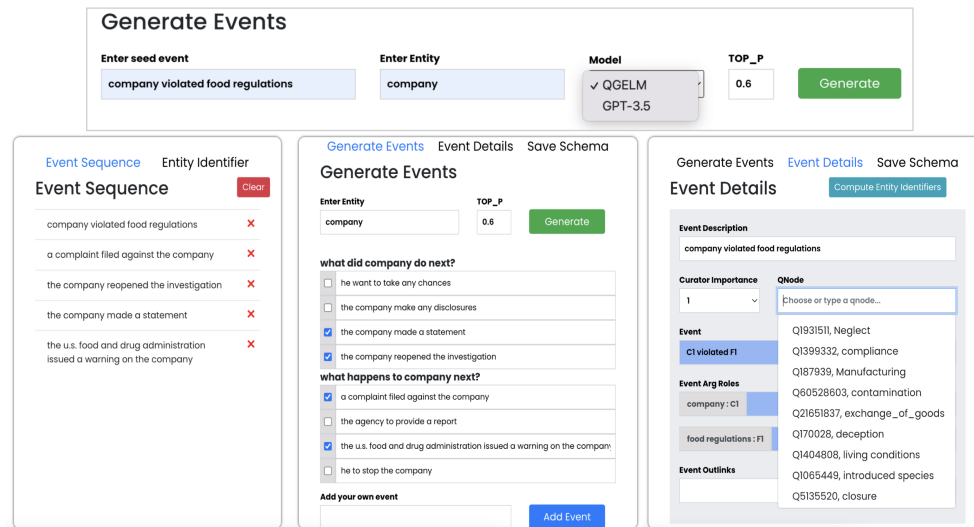


Figure 3: Curator section of SAGEViz user interface. This part of the UI is used to generate event sequences using different models that can be selected from the drop-down menu. Events generation, entity coreference resolution, etc. are all done on the curator section.

**Event Role Labeler** An event generated by our event language model consists of a subject-verb-object (SVO) format. Identifying the semantic roles of the entities enriches schematic knowledge regarding it participants. This version of SAGEViz uses manual assignment of semantic roles to entities (as existing systems do not perform well on events tuple representations) but the system design supports replacing it with an automatic system.

**Q-node Linker** Using the SVO formatted events, we use the entity linking model and python library BLINK (Wu et al., 2020) to identify Q-nodes for the verb (to represent the event), and Q-nodes for entities in both the subject and object. Since an entity or event could be associated with multiple Q-nodes, users can search for a Q-node from a provided list.

**Schema Visualizer** Once the schema is saved, all the information about the events, their relationships and the participating entities are converted to JSON format. This file is parsed into a directed graph structure, which is then ingested, visualized and presented in the visualizer, as shown in Figure 1.

### 2.3 User Interface

Multiple users can interact with SAGEViz (and its different components) simultaneously and curate or visualize schemas through an easy-to-use interface. Figure 2 shows the main interface of the system with its core components (curator and visualizer).

**Curator** Choosing the curator button will take users to another page (shown in Figure 3) where they can start with the event generator component. They can specify a seed event and an entity of interest (shown in the top section of Figure 3) and then the system provides users with a list of events specific to that entity (Generate Events box in the middle section of Figure 3). Now, the user has the option of selecting as many relevant events as they desire, edit the generated events, or add an event which they think might fit in the sequence to guide the next generation steps of the model. The events will be added to the Event Sequence box (left side of Figure 3) as the user selects them.

Once the user is done generating events for a seed, they can use the Entity identifier tab of the Event Sequence box to ask the system to identify the corefering entities and assign unique ids to them (as described earlier).

Finally, the EventDetails box (Figure 3) is where the users can search the Qnodes of the events as well as assign importance scores to the events. Once the user is satisfied with the generated sequence and after filling in all the necessary details, they can use the Save Schema tab to save it.

**Visualizer** The Visualizer component of the UI can represent the generated schemas in a multi-level graph representation. Once you click on Visualizer button, you will see the high-level graph structure of the (disease outbreak) schema containing the high-level stages as shown in Fig-

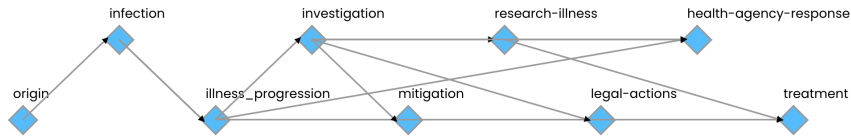


Figure 4: A high level view of the Disease Outbreak schema showing the first-level stages or sub-schemas. These sub-schemas can further be expanded into other sub-schemas or primitive events.

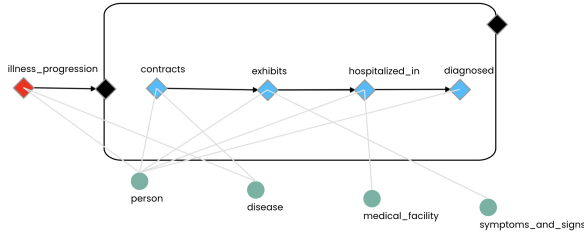


Figure 5: An expanded view of the illness\_progression sub-schema. This is a sub-schema within the disease outbreak schema shown in Figure 4.

ure 4 and the temporal relations between these stages. You can further click on each node and then you can see the fine-grained representation of each stage consisting of all the events, entities, and the relations between entities and events (shown in Figure 5). You can keep getting deeper into the representation for all non-primitive events (primitive events are the atomic events that are not expanded).

## 2.4 Implementation Details

We now describe the technical details of the web application and how it is built in detail.

**Frontend** The frontend application uses React, a JavaScript library for creating user interfaces with modular components. To maintain visual consistency, React-Bootstrap is employed, providing a set of commonly used components. We utilize Webpack to generate static assets for the application and Express.js to serve the application. A significant challenge we encountered was optimizing the bundle size to enhance the application’s performance and overall user experience. To address this, we focused on developing reusable components, which allowed us to reduce redundancy and improve efficiency. Additionally, we leveraged the browser’s built-in caching capabilities to cache static assets, resulting in faster subsequent visits and reduced server load. Finally, we deployed the application on Amazon EC2 (EC2, 2023) which offers a robust, scalable, and cost-effective infrastructure.

**Backend** The backend APIs are developed using Flask (Grinberg, 2018), a lightweight Python web framework, which makes it an ideal choice

for hosting our Question-guided event language model. We load the language model (Koupaee et al., 2023) and SpanBERT model (for coreference resolution) during application startup. For the GPT-3.5 few-shot event generator we use the OpenAI python library which gives us access to the gpt-3.5-turbo model. To optimize the usage of the gpt-3.5-turbo model and enhance overall performance, we implement a response caching mechanism. This caching system stores the responses obtained from the OpenAI API (OpenAI, 2023), preventing the need for repeated calls for the same or similar queries. By doing so, we reduce the number of API requests and consequently, minimize costs and latency. The caching strategy efficiently handles recurrent queries and ensures that if an identical request is made in the future, the corresponding response can be retrieved from the cache, avoiding unnecessary re-computation.

## 3 User Study

We aimed to study the effectiveness and efficiency of SAGEViz at generating schemas using the process outlined in Koupaee et al. (2023) and compared it with the schema generation interactive system (CLI) used in this study. For a fair comparison, we selected only the event generator component of SAGEViz. We followed the same guidelines and used the same test data consisting of 35 seeds from 8 domains. Four different users interacted with the system spending 4 minutes on each seed, as was done in the previous study, selecting and adding events to the schema that are *sensible*, *relevant*, *unique* and *typical* for the schema context.

For each 4-minute interaction, a user started with a seed event as the schema context and a participating entity to use in the question to the event generator, and generated events using the Question Guided Event Language Model (QGELM) (Koupaee et al., 2023). From the generated events, the user selected and added each event that satisfied all 4 of the selection criteria (each is a binary judgment). If any of the events was selected and added, the generation step is counted as an accepted



| Metric             | CLI  | SAGEViz     |
|--------------------|------|-------------|
| # events ▲         | 8.8  | <b>12.1</b> |
| % accepted steps ▲ | 73.3 | <b>80.0</b> |
| % rejected steps ▼ | 26.7 | <b>20.0</b> |
| total steps        | 12.0 | 7.48        |

Table 1: Quantitative analysis of schema generation using the our tool and the baseline. The higher the average the better a system is for metrics with ▲ whereas lower values are desired for metrics with ▼.

step and if none of the events fit the criterion, no events are added, and the step is counted as a rejected step before the user generates for the next step. Generation is stopped when the 4 minutes elapse or if no events that fit the selection criteria can be found in the generations. The participating entity (in the question to the event generator) can be changed in any step to encourage the generation of diverse events that fit the selection criteria. As shown in Table 1, SAGEViz allows a user to generate schemas more effectively and efficiently when compared to CLI (Koupae et al., 2023). On average, SAGEViz, allows a user to select more events (# events) with fewer interactions (total steps). The capability to edit generated events in SAGEViz leads to fewer regenerations (% rejected steps) with more generation steps (% accepted steps) accepted by users.

## 4 Related Work

### 4.1 Schema Induction

Chambers and Jurafsky (2008, 2009) automatically learned a schema from newswire text based on coreference and statistical probability models. Peng and Roth (2016); Peng et al. (2019) generated an event schema based on their proposed semantic language model., representing the whole schema as a linear sequence of abstract VerbNet (Schuler, 2005) verb senses. In these works, the schema was created for a single actor (protagonist). It caused limited coverage in a more complex scenario. Further, 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 treated 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 which by-

passed the event retrieval and extraction process and solved the domain transfer problem, but, it suffered from the incompleteness and instability of GPT-3 outputs. Currently, there is neither a perfect solution for schema induction without manual post-processing, nor a human correction system (Du et al., 2022). Our demonstration system develops a curation interface that can generate a comprehensive schema with a human curator in the loop.

### 4.2 Human-in-the-loop Schema Curation Interface

Another related area is the human-in-the-loop schema generation, where annotators collaborate with computational models to create a high-quality event schema. Machine-Assisted Script Curation (Ciosici et al., 2021) was 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 to other generative models. Our interface makes use of pre-trained LLMs to automatically generate schema content, compared to their interface which largely counts on human input. 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.

## 5 Conclusion

Prior work on schema induction either relied on existing information extraction pipelines to convert unstructured documents into event graphs, or required massive human effort to annotate event schemas. The former lacks the guarantee of being high-quality, while the latter is hard to scale due to its demands of time, effort and cost. We alleviated these problems with a web application that leverages the power of LLMs and the reliability provided by human intervention to create, expand, and visualize event schemas. Our tool enables users to collaboratively and seamlessly generate and update

event sequences. We believe that our human-in-the-loop tool reduces expert effort for creating new schemas and analyzing complex event sequences.

## Acknowledgments

We thank the anonymous reviewers for their insightful feedback and suggestions. This material is based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes.

## References

- Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731.
- 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.
- Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.
- 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.
- 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.
- AWS EC2. 2023. Amazon elastic compute cloud (amazon ec2) provides on-demand, scalable computing capacity in the amazon web services (aws) cloud. <https://aws.amazon.com/ec2/>.
- Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc."
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Heng Ji and Ralph Grishman. 2008. [Refining event extraction through cross-document inference](#). In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. [Spanbert: Improving pre-training by representing and predicting spans](#). *CoRR*, abs/1907.10529.
- Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2021. Don't let discourse confine your model: Sequence perturbations for improved event 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 2: Short Papers)*, pages 599–604.
- Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2023. [Modeling complex event scenarios via simple entity-focused questions](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2468–2483, Dubrovnik, Croatia. Association for Computational Linguistics.
- Heeyoung Kwon, Mahnaz Koupaee, Pratyush Singh, Gargi Sawhney, Anmol Shukla, Keerthi Kumar Kallur, Nathanael Chambers, and Niranjan Balasubramanian. 2020. Modeling preconditions in text with a crowd-sourced dataset. *arXiv preprint arXiv:2010.02429*.

- 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.
- Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. [In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval](#). In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online. Association for Computational Linguistics.
- Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*, pages 159–164.
- 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.
- Raymond J Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*, pages 681–687.
- OpenAI. 2023. Openai api provides a general-purpose “text in, text out” interface. <https://platform.openai.com/overview>.
- 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*.
- Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.
- Mehdi Rezaee, Francis Ferraro, et al. 2021. Event representation with sequential, semi-supervised discrete variables. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI’75*, page 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.
- Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Hierarchical quantized representations for script generation. *arXiv preprint arXiv:1808.09542*.
- Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

# Thresh 🌿: A Unified, Customizable and Deployable Platform for Fine-Grained Text Evaluation

David Heineman, Yao Dou, Wei Xu

School of Interactive Computing, Georgia Institute of Technology

{david.heineman, douy}@gatech.edu; wei.xu@cc.gatech.edu

## Abstract

Fine-grained, span-level human evaluation has emerged as a reliable and robust method for evaluating text generation tasks such as summarization, simplification, machine translation and news generation, and the derived annotations have been useful for training automatic metrics and improving language models. However, existing annotation tools implemented for these evaluation frameworks lack the adaptability to be extended to different domains or languages, or modify annotation settings according to user needs; and, the absence of a unified annotated data format inhibits the research in multi-task learning. In this paper, we introduce Thresh 🌿, a unified, customizable and deployable platform for fine-grained evaluation. With a single YAML configuration file, users can build and test an annotation interface for any framework within minutes – all in one web browser window. To facilitate collaboration and sharing, Thresh provides a community hub that hosts a collection of fine-grained frameworks and corresponding annotations made and collected by the community, covering a wide range of NLP tasks. For deployment, Thresh offers multiple options for any scale of annotation projects from small manual inspections to large crowdsourcing ones. Additionally, we introduce a Python library to streamline the entire process from typology design and deployment to annotation processing. Thresh is publicly accessible at <https://thresh.tools>.

## 1 Introduction

As modern large language models are able to generate human-level quality text (Brown et al., 2020; OpenAI, 2023), the evaluation of these models becomes increasingly challenging. Recent work has shown traditional surface-level evaluation methods such as pairwise comparison or Likert-scale ratings become less reliable (Clark et al., 2021; Maddela et al., 2023) due to the close performance of these LLMs. To address this, several fine-grained human

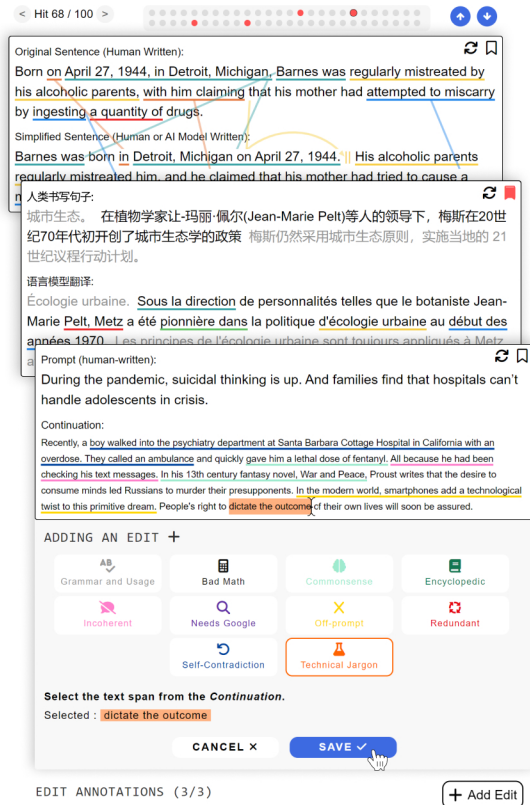


Figure 1: Examples of fine-grained evaluation frameworks implemented on Thresh. In order: SALSA (Heineman et al., 2023), MQM (Freitag et al., 2021), Scarecrow (Dou et al., 2022a).

evaluation frameworks have been proposed for various tasks such as open-ended generation (Dou et al., 2022a), text simplification (Heineman et al., 2023), and machine translation (Freitag et al., 2021). In these frameworks, annotators identify and annotate specific spans corresponding to quality or errors in the generated text.

However, each of these evaluation frameworks releases its own dedicated annotation interface that is difficult to modify or adapt to different evaluation schemes, thus limiting the customizability. For example, Scarecrow’s typology (Dou et al., 2022a), which is designed for open-ended text generation

| Framework                                | Task                               | Released |
|--|------------------------------------|----------|
| <i>Evaluation</i>                        |                                    |          |
| MQM (Freitag et al., 2021)               | Translation                        | ✓        |
| FRANK (Pagnoni et al., 2021)             | Summarization                      | ✓        |
| SNaC (Goyal et al., 2022b)               | Narrative Summarization            | ✓        |
| Scarecrow (Dou et al., 2022a)            | Open-ended Generation              | ✓        |
| SALSA (Heineman et al., 2023)            | Simplification                     | ✓        |
| ERRANT (Bryant et al., 2017)             | Grammar Error Correction           | ✗        |
| FG-RLHF (Wu et al., 2023)                | Fine-Grained RLHF                  | ✓        |
| <i>Inspection</i>                        |                                    |          |
| MultiPIT (Dou et al., 2022b)             | Paraphrase Generation              | ✗        |
| CWZCC (Himoro and Pareja-Lora, 2020)     | Zamboanga Chavacano Spell Checking | ✗        |
| Propaganda (Da San Martino et al., 2019) | Propaganda Analysis                | ✓        |
| arXivEdits (Jiang et al., 2022)          | Scientific Text Revision           | ✓        |

Table 1: Existing typologies currently implemented on Thresh. *Released* indicates whether the annotated data is released. Corresponding links on Thresh for each framework can be found in Table 2 in the Appendix.

for news, may require modifications when applied to other domains such as story or scientific writing. Frameworks like MQM (Freitag et al., 2021) only allow selections of the spans in the target sentence, restricting the ability to select the associated source spans in error categories such as mistranslation. Furthermore, modern LLMs are ideally evaluated on multiple tasks (Hendrycks et al., 2021), but the lack of a unified annotation tool makes this process inconvenient. Considering the recent success of multi-task instruction fine-tuning (Wei et al., 2021; Sanh et al., 2021), a standardized annotation format would enable research in multi-task learning with fine-grained human feedback.

To this end, we present Thresh 🌟: a unified and customizable platform for building, distributing and orchestrating fine-grained human evaluation for text generation in an efficient and easy-to-use manner. Our platform allows users to create, test and deploy an evaluation framework within minutes, all in a single browser window and has already been used to orchestrate large-scale data annotation (Heineman et al., 2023). Thresh also serves as a *community hub* for fine-grained evaluation frameworks and annotation data, all presented in a unified format. Figure 1 displays three examples of evaluation frameworks built on Thresh. The following are the design principles of Thresh:

- **Unified:** Thresh standardizes fine-grained evaluation into two key components: span selection and span annotation. Users can easily implement any framework by writing a YAML template file (see Figure 5), and Thresh will build the corresponding annotation interface. All resulting annotations adhere to a consistent JSON format.

- **Customizable:** Thresh offers extensive customization to meet a wide range of user needs. This includes different span selection methods from subword to word-level, diverse annotation options including custom questions and text boxes to handle arbitrary typologies, as well as customized interface elements in any language.
- **Deployable:** Thresh supports a range of deployment options for annotation projects of various scales. Small-scale linguistic inspections (e.g., manual ablation studies) can be directly hosted on the platform. For larger projects, users can host their template in a GitHub repository and connect to Thresh. Thresh is also compatible with crowdsourcing platforms such as Prolific<sup>1</sup> and Amazon MTurk<sup>2</sup>.
- **Contributive:** Thresh also operates as a community hub where users can contribute and access a wide variety of fine-grained evaluation frameworks and their annotation data. Currently, it includes 11 frameworks as displayed in Table 1.
- **End-to-End:** Beyond facilitating the creation and deployment of evaluation frameworks, Thresh streamlines every step of the annotation process. It offers functions for authors to publish their typologies as research artifacts and a supplementary Python library, released under the Apache 2.0 license, to help data collection.<sup>3</sup>

## 2 Related Work

**Fine-grained Text Evaluation.** Given the limitations of traditional human evaluation methods such as Likert-scale and pairwise comparison in the era of LLMs, many recent studies have proposed fine-grained human evaluation frameworks. Dou et al. (2022a) introduces Scarecrow to capture error spans in open-ended text generation for news, MQM (Freitag et al., 2021) identifies errors in machine translation, and FRANK (Pagnoni et al., 2021) captures factual errors in abstractive text summarization. We list other evaluation and inspection typologies in Table 1. However, these existing frameworks usually develop their own annotation tools which lack customizability and universality, making them difficult to adapt to other languages or domains, or to new annotation settings. Recently, Goyal et al. (2022a) proposes

<sup>1</sup><https://www.prolific.co>

<sup>2</sup><https://www.mturk.com>

<sup>3</sup><https://www.pypi.org/project/thresh>

FALTE, customizable span-level error highlighting for long text evaluation, but it only includes a subset of features offered by Thresh, limiting its ability to implement complex typologies such as SALSA (Heineman et al., 2023). Specifically, FALTE only highlights errors without rating their severity or efficacy, does not support multi-span or composite selection, and cannot select overlapping spans. Moreover, its lack of a tree structure can make the interface cluttered if there are more than a handful of categories. Thresh instead builds unified and customizable support across task setups.

**Annotation Tool.** Accessible and replicable annotation tools have been a persistent goal for NLP tasks. Stenetorp et al. (2012) introduces BRAT, the first web browser-based annotation tool and Yimam et al. (2013) further improves BRAT on speed and label configuration. In recent years, a new generation of universal annotation tools have been introduced by academia and industry, including Prodigy (Montani and Honnibal, 2018), Doccano (Nakayama et al., 2018), LightTag (Perry, 2021), and POTATO (Pei et al., 2022). Focusing on universality, these tools allow authors to add custom UI elements such as multiple choice questions, text boxes or pairwise comparison. However, these surface-level annotation options are not sufficient to implement complex typology setups demanded by fine-grained evaluation, which are typically structured by decision trees (Heineman et al., 2023). Thresh addresses this gap by recursively building the interface, which allows for nested questions. Besides, Thresh encourages sharing and reproducibility by providing a community hub where users can upload their new or use existing fine-grained frameworks and annotated data.

**Span-level Annotation.** Span-level annotation has a long history across NLP tasks. In Named Entity Recognition (NER), spans are selected and labeled as names of persons, organizations, locations, or other entities (Tjong Kim Sang and De Meulder, 2003). Word alignment focuses on selecting aligned words or phrases between two parallel corpora across languages (Och and Ney, 2003), or within monolingual tasks (Lan et al., 2021). Span selection has also been used for question answering such as in SQuAD (Rajpurkar et al., 2016), where the answer is defined by a span within the document context. Furthermore, extractive text summarization (Hermann et al., 2015) highlights

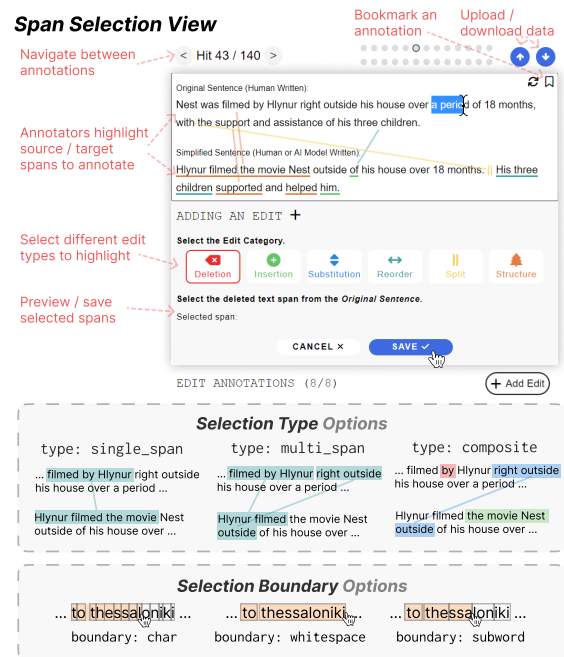


Figure 2: The span selection component of Thresh, customized with the SALSA (Heineman et al., 2023) typology as an example.

the spans that summarizes a given document. With a goal of understanding where and how text generation succeeds or fails, fine-grained text evaluation selects spans that are either quality or error in generated text. These selected spans are then annotated following a complex typology and rated on the severity of errors or efficacy of high-quality content (Freitag et al., 2021; Dou et al., 2022a; Heineman et al., 2023).

### 3 Fine-Grained Text Evaluation

Thresh formulates fine-grained text evaluation as two components: *span selection* and *span annotation*. During development, users define their annotation typology and interface features using a YAML template (see Sec 4 and Fig 5 for more details). Based on the configuration, Thresh then constructs an annotation interface that integrates both components, as illustrated in Figures 2 and 3.

#### 3.1 Span Selection

Each annotation instance consists of the *source*, *target* and *context*. For example, in open-ended text generation (Zellers et al., 2019), the source is a starting sentence and the target is a model-generated continuation. In text simplification (Xu et al., 2016), the source would be a complex sentence or paragraph, and the target would be the generated simplification. The context holds additional relevant information, such as a prompt instruction,

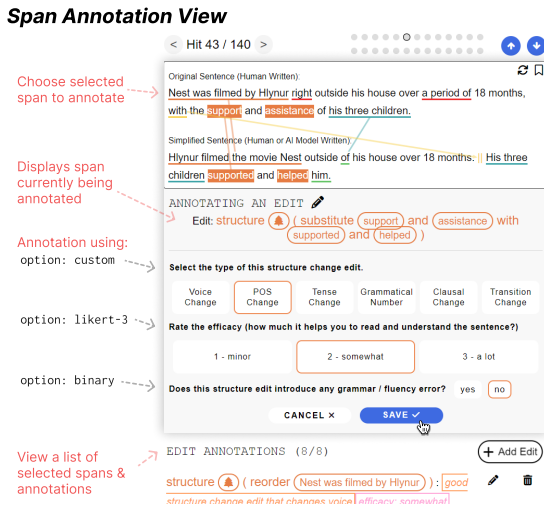


Figure 3: The span annotation component of Thresh, customized with the SALSA (Heineman et al., 2023) typology as an example.

a retrieved Wikipedia page, or a dialogue history. During the span selection stage, annotators select relevant spans, referred to as *Edits*, in the source and target, following the edit category definitions outlined in the typology, as illustrated in Figure 2.

**Selection Type.** For each edit category, users can specify one of three selection types: *single-span*, *multi-span*, or *composite* – the latter grouping together multiple single-span or multi-span selections. Multi-span selection is well-suited for edits that impact multiple parts of the source or target, e.g., the “Redundant” error in Scarecrow (Dou et al., 2022a), which requires selecting both the repetitive spans and their antecedents. Composite selections are ideal for high-level edits performed as a combination of several low-level edits, e.g., the “Structure” edit in SALSA (Heineman et al., 2023). Users can also customize each edit category to be selectable not only on the target, but also on the source (e.g., “Deletion” edit), or on both (e.g., “Substitution” edit), useful for text revision tasks.

**Selection Boundary.** Many span-selection interfaces define selection boundaries as each character, which can inadvertently lead to partial word selections and slow the annotation process. Dou et al. (2022a) proposes a solution that “snaps” the selection to the nearest whitespace, but this approach is limited in: (1) punctuation gets selected with adjacent words, even when this is not intended by annotators, (2) languages with no whitespace boundaries between words (e.g., Chinese) cannot be supported and (3) the annotation data cannot be perfectly translated to training data for token-level

labeling tasks. We therefore introduce sub-word boundaries as a third option, in which users can use any LLMs tokenizer of their choice (such as RobertaTokenizer from Transformers<sup>4</sup>) to tokenize the data and specify a boundary: subword flag in the YAML configuration file.

### 3.2 Span Annotation

In the YAML file, users define the typology in a decision tree structure to further categorize the selected spans into fine-grained types. Unlike previous work which presents all fine-grained edit types to annotators simultaneously, Thresh recursively compiles the annotation interface. Annotators thus will answer a series of questions or follow-up questions under each edit type, as shown in Figure 3. This tree structure enables support for complex error typologies. An example of this can be seen in Figure 4, which shows a 35-category typology implementation for a grammar error correction task. Thresh supports binary, three and five-scale questions with customized label names, as well as text boxes for tasks that require human post-editing or explanations. With these features, our interface supports complex annotation schemes in a flexible and easily extensible way.

We also give users the option of only enabling one of the two above components. This allows annotation for word/span alignment tasks (Sultan et al., 2014) (where no annotation is needed) or two-stage annotation, where one set of annotators selects spans and then another set labels them.

### 3.3 Additional Features

**Adjudication View.** Using the adjudication flag, users can deploy two or three interfaces side-by-side, allowing adjudicators to inspect annotators’ quality by comparing multiple candidate annotations simultaneously.

**Multi-Language Support.** Fine-grained evaluation has seen almost exclusive attention to English tasks (Huidrom and Belz, 2022). To smoothen the deployment barrier for multilingual fine-grained evaluation, all interface elements can be overridden to suit any language. For our default interface text, we support 14 translations which can be enabled out-of-the-box by adding a language flag: *zh*, *en*, *es*, *hi*, *pt*, *bn*, *ru*, *ja*, *vi*, *tr*, *ko*, *fr* and *ur*.

**Instructions.** Users may write interface instructions with Markdown formatting, which allows for

<sup>4</sup><https://www.github.com/huggingface/tokenizers>



Figure 4: The left figure shows a grammar error typology with 35 categories for contemporary written Zamboangueño Chabacano, a variant of Philippine Creole Spanish (Himoro and Pareja-Lora, 2020). The center figure shows its annotation interface built on Thresh, highlighting the ability for Thresh to support complex, recursive annotation trees. The right figure shows the Python serialization for the annotation, generated by the Thresh library.

links, pictures and inline code. They have the option to display their instructions as a pop-up modal, or prepend the text above the interface.

**Paragraph-level Annotation.** By breaking evaluation down to individual sentences, authors can reduce the cognitive load required for lengthy annotation tasks such as identifying errors in long-form summarization (Goyal et al., 2022a). Users can specify an additional `context_before` or `context_after` field to add paragraph-level context or custom display options to view paragraphs text side-by-side with selected edits.

#### 4 Interactive Interface Builder

To alleviate the time consuming process of customizing and hosting front-end code — even building custom databases in some cases — Thresh implements an in-browser interface builder, which allows users to create, test and deploy a fine-grained interface within a single web browser page, as depicted in Figure 5. Users write a YAML template to construct their interface and provide data with a JSON file. The *Compile* button allows users to preview their interface, and the *Deploy* button presents instructions for different deployment options, which are described in §5.

**Template Hub.** As Thresh aims to facilitate easy use and distribution of fine-grained evaluation frameworks, it provides a template hub that makes it simple for any NLP practitioner to access a framework with their own data. Alongside the 10 tutorial templates that explain each interface feature, the annotation builder currently includes 11 widely used inspection and evaluation typologies

across major text generation tasks. Table 1 (on Page 2) lists each framework, its associated task and link to our implementation.

To upload a framework to Thresh, users can create a GitHub pull request with their typology’s YAML file, which is merged publicly. We also include other features to facilitate sharing and replication. Users can add a citation flag along with a BibTeX citation, which creates a *Cite this Typology* button in the annotation builder, a `paper_link` flag, which adds a link to their research paper in the builder and on deployment, and a `demo_data_link` flag which creates a *View Demo Data* button to allow viewers to use the interface with example data.

For testing, users can paste data into the interface builder interactively, and for deployment can link to data files. Data can be blank or come with existing annotations, in which case the annotations will be appropriately parsed, verified and rendered.

**Unified Data Model.** As shown in Table 1 on Page 2, many existing frameworks have released their annotated data, but in varied formats. To ensure compatibility, we create conversion scripts that adapt these annotations to our unified format. Our scripts are designed to be *bidirectional*, meaning data published for these typologies can be converted to our format and back without data loss. Our unified fine-grained data format allows smooth transfer of analysis, agreement calculation and modeling code between different projects. We believe this will support research in learning with multi-task fine-grained training setups or model feedback. Like framework templates, users can upload their annotated data to the hub via a GitHub pull request.



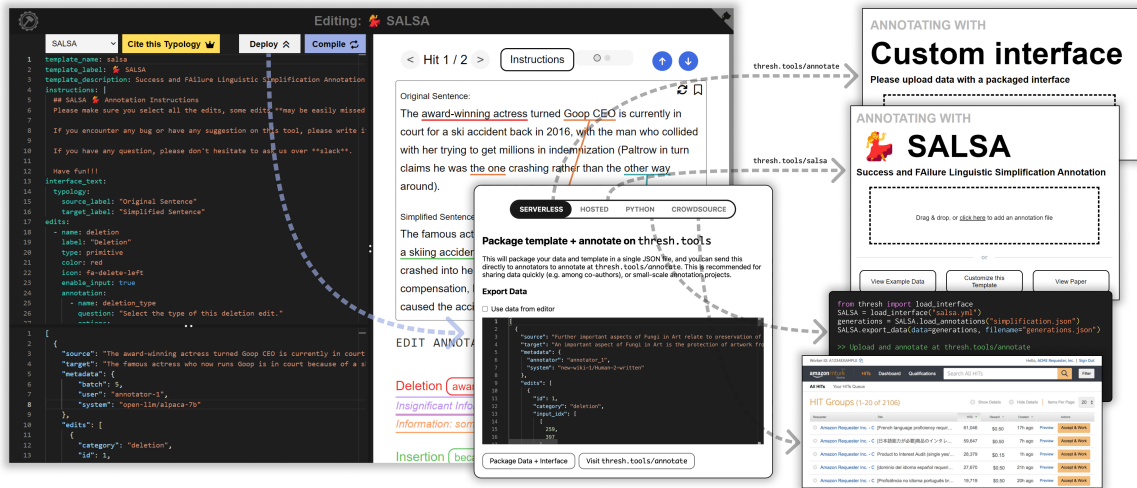


Figure 5: Thresh deployment workflow. Users build and test their template and then deploy with one of 4 options.

## 5 Deployment

Managing and collecting fine-grained annotations becomes bulky at scale, we thus release supplementary tools to deploy interfaces quickly or programmatically, and integrate loading annotations directly into Python. This includes the `thresh` library<sup>5</sup>, which is useful for compiling interfaces and loading annotations. We support the following deployment types as shown in Figure 5:

- **Hosted:** Best for small-scale inspection or data exploration, users can download a file that bundles the data and template together. Then, users can upload this file to `thresh.tools/annotate` to begin annotation immediately.
- **Serverless:** Users upload their YAML template to a public repository such as GitHub or HuggingFace, and link their template to `thresh.tools` through a URL parameter: `gh` or `hf` respectively. Users can also link data via the `d` parameter. In addition, we release demo code for users to host their interface on their own domain without cloning the Thresh repository.
- **Python:** For large scale projects, users can programmatically generate and deploy templates using the `create_template` functionality provided in the `thresh` library. This helps for projects with a large number of templates, such as annotation in multiple languages. Additionally, integration with Python allows a direct connection from model generation to annotation processing, supporting the creation of workflows like fine-grained RLHF (Wu et al., 2023).
- **Crowdsourcing:** If the data collection process is

mishandled, annotation by crowdworkers can lead to poorly standardized or noisy data (Karpinska et al., 2021; Veselovsky et al., 2023). To assist annotation quality control, we publish tools to encourage best practices when using crowdsourcing platforms. Our crowdsourcing deployment workflow includes example code for interactive, multi-stage tutorials to create qualification tasks and step-by-step tutorials for deployment on both Prolific and Amazon Mechanical Turk.

Additionally, we support lightweight database integration (such as with Google Firebase<sup>6</sup>) for all deployment types, allowing users to connect their own database to any annotation setup.

**Python Serialization.** Compared to previous work that simply exports JSON annotations, our supplementary `thresh` library includes functionality for loading and combining annotation files to simplify the data ingestion process. For example, `load_annotations` merges multiple data files, serializes the data into Python objects, and evaluates whether the data collected is consistent with the configuration used to load the data.

## 6 Conclusion

We present Thresh 🌱, a unified, customizable, and deployable platform for fine-grained text evaluation. Thresh offers extensive customization via a simple YAML configuration file, and facilitates a community hub for sharing frameworks and annotations. The platform also ensures seamless deployment for any scale of annotation projects and introduces a Python library to further ease the process from typology design to annotation processing.

<sup>5</sup><https://www.pypi.org/project/thresh>

<sup>6</sup><https://firebase.google.com>

## Ethical Considerations

We do not anticipate any ethical issues pertaining to the topics of fine-grained evaluation supported by our interface. Nevertheless, as Thresh lowers the barrier to fine-grained evaluation, vast ethical responsibility falls upon practitioners using our platform to prevent the exploitation of crowdsourced workers, through fair pay (Fort et al., 2011) and safeguards against exposure to harmful or unethical content (Shmueli et al., 2021). As task difficulty and complexity scales with the granularity of data collected, increasing care must be taken for training annotators adequately and to scale pay accordingly (Williams et al., 2019).

## Acknowledgments

This research is supported in part by the NSF awards IIS-2144493 and IIS-2112633, ODNI and IARPA via the HIATUS program (contract 2022-22072200004). 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 NSF, 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

- 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, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff 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. *ArXiv*, abs/2005.14165.
- Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. [Automatic annotation and evaluation of error types for grammatical error correction](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.
- Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. [All that’s ‘human’ is not gold: Evaluating human evaluation of generated text](#). 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 7282–7296, Online. Association for Computational Linguistics.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.
- Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2022a. [Is GPT-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland. Association for Computational Linguistics.
- Yao Dou, Chao Jiang, and Wei Xu. 2022b. [Improving large-scale paraphrase acquisition and generation](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9301–9323, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Karèn Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. [Last words: Amazon Mechanical Turk: Gold mine or coal mine?](#) *Computational Linguistics*, 37(2):413–420.
- Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. [Experts, errors, and context: A large-scale study of human evaluation for machine translation](#). *Transactions of the Association for Computational Linguistics*, 9:1460–1474.
- Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022a. [FALTE: A toolkit for fine-grained annotation for long text evaluation](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–358, Abu Dhabi, UAE. Association for Computational Linguistics.
- Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022b. [SNaC: Coherence error detection for narrative summarization](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 444–463, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- David Heineman, Yao Dou, Mounica Maddela, and Wei Xu. 2023. [Dancing between success and failure: Edit-level simplification evaluation using SALSA](#). *arXiv preprint arXiv:2305.14458*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.

2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.
- Marcelo Yuji Himoro and Antonio Pareja-Lora. 2020. [Towards a spell checker for Zamboanga Chavacano orthography](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2685–2697, Marseille, France. European Language Resources Association.
- Rudali Huidrom and Anya Belz. 2022. [A survey of recent error annotation schemes for automatically generated text](#). In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 383–398, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Chao Jiang, Wei Xu, and Samuel Stevens. 2022. [arXivEdits: Understanding the human revision process in scientific writing](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9420–9435, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. [The perils of using Mechanical Turk to evaluate open-ended text generation](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Wuwei Lan, Chao Jiang, and Wei Xu. 2021. [Neural semi-Markov CRF for monolingual word alignment](#). 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 6815–6828, Online. Association for Computational Linguistics.
- Mounica Maddela, Yao Dou, David Heineman, and Wei Xu. 2023. [LENS: A learnable evaluation metric for text simplification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.
- Ines Montani and Matthew Honnibal. 2018. [Prodigy: A modern and scriptable annotation tool for creating training data for machine learning models](#).
- 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>.
- Franz Josef Och and Hermann Ney. 2003. [A systematic comparison of various statistical alignment models](#). *Computational Linguistics*, 29(1):19–51.
- OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.
- Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. [Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, Online. 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 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.
- 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.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.
- Boaz Shmueli, Jan Fell, Soumya Ray, and Lun-Wei Ku. 2021. [Beyond fair pay: Ethical implications of NLP crowdsourcing](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3758–3769, Online. 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.
- Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment:

- Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. [Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition](#). In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. 2023. Artificial artificial artificial intelligence: Crowd workers widely use large language models for text production tasks. *arXiv preprint arXiv:2306.07899*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Alex C Williams, Gloria Mark, Kristy Milland, Edward Lank, and Edith Law. 2019. The perpetual work life of crowdworkers: How tooling practices increase fragmentation in crowdwork. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–28.
- Zequi Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*.
- Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. [Optimizing statistical machine translation for text simplification](#). *Transactions of the Association for Computational Linguistics*, 4:401–415.
- 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, Sofia, Bulgaria. Association for Computational Linguistics.
- Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

| Framework                                | Task                               | Released | Link  |
|--|------------------------------------|----------|---|
| <i>Evaluation</i>                        |                                    |          |   |
| MQM (Freitag et al., 2021)               | Translation                        | ✓        | <a href="https://thresh.tools/mqm">thresh.tools/mqm</a>               |
| FRANK (Pagnoni et al., 2021)             | Summarization                      | ✓        | <a href="https://thresh.tools/frank">thresh.tools/frank</a>           |
| SNaC (Goyal et al., 2022b)               | Narrative Summarization            | ✓        | <a href="https://thresh.tools/snac">thresh.tools/snac</a>             |
| Scarecrow (Dou et al., 2022a)            | Open-ended Generation              | ✓        | <a href="https://thresh.tools/scarecrow">thresh.tools/scarecrow</a>   |
| SALSA (Heineman et al., 2023)            | Simplification                     | ✓        | <a href="https://thresh.tools/salsa">thresh.tools/salsa</a>           |
| ERRANT (Bryant et al., 2017)             | Grammar Error Correction           | ✗        | <a href="https://thresh.tools/errant">thresh.tools/errant</a>         |
| FG-RLHF (Wu et al., 2023)                | Fine-Grained RLHF                  | ✓        | <a href="https://thresh.tools/fg-rlhf">thresh.tools/fg-rlhf</a>       |
| <i>Inspection</i>                        |                                    |          |   |
| MultiPIT (Dou et al., 2022b)             | Paraphrase Generation              | ✗        | <a href="https://thresh.tools/multipit">thresh.tools/multipit</a>     |
| CWZCC (Himoro and Pareja-Lora, 2020)     | Zamboanga Chavacano Spell Checking | ✗        | <a href="https://thresh.tools/cwzcc">thresh.tools/cwzcc</a>           |
| Propaganda (Da San Martino et al., 2019) | Propaganda Analysis                | ✓        | <a href="https://thresh.tools/propaganda">thresh.tools/propaganda</a> |
| arXivEdits (Jiang et al., 2022)          | Scientific Text Revision           | ✓        | <a href="https://thresh.tools/arxivedits">thresh.tools/arxivedits</a> |

Table 2: Existing typologies implemented on Thresh with their associated link. *Released* indicates whether the annotated data is released.

# InsightPilot: An LLM-Empowered Automated Data Exploration System

Pingchuan Ma  
HKUST

Rui Ding<sup>†</sup>  
Microsoft Research

Shuai Wang<sup>†</sup>  
HKUST

Shi Han  
Microsoft Research

Dongmei Zhang  
Microsoft Research

## Abstract

Exploring data is crucial in data analysis, as it helps users understand and interpret the data more effectively. However, performing effective data exploration requires in-depth knowledge of the dataset, the user intent and expertise in data analysis techniques. Not being familiar with either can create obstacles that make the process time-consuming and overwhelming.

To address this issue, we introduce *InsightPilot*, an LLM (Large Language Model)-based, automated data exploration system designed to simplify the data exploration process. *InsightPilot* features a set of carefully designed analysis actions that streamline the data exploration process. Given a natural language question, *InsightPilot* collaborates with the LLM to issue a sequence of analysis actions, explore the data and generate insights. We demonstrate the effectiveness of *InsightPilot* in a user study and a case study, showing how it can help users gain valuable insights from their datasets.

## 1 Introduction

Exploratory data analysis (EDA) is a demanding task that extracts meaningful insights from data (Komorowski et al., 2016; Jebb et al., 2017; Devore, 2007). Data exploration is a critical step in data analysis. In general, it involves a series of data analysis operations, such as filtering, sorting, and grouping, to discover patterns in data. Usually, the process is iterative and interactive, and the user needs to manually explore the data back-and-forth to gain insights. This process is often time-consuming and requires considerable domain knowledge and expertise. Below, we present an example to illustrate a data exploration process.

**Example.** *Using a student performance dataset from multiple schools (Figure 1), an education analyst, Alice, conducts EDA to comprehend trends in math performance. After considerable manual data filtering and sorting, Alice captures an upward*

*trend by plotting math scores over time. Alice then puts in more effort into manual data filtering for comparing student performance across schools A, B, and C. Finally, she observes that both schools A and B illustrate an increasing trend while school C has an outlier in 2020. Alice is curious and decides to investigate the outlier. She spends even more time exploring the data back-and-forth, filtering and grouping by various variables until she finally finds that when excluding “take-home” exams, the outlier for school C in 2020 is no longer present. Alice notes this finding and concludes that the outlier is caused by a policy change of the exam form in school C in 2020.*

Alice’s manual data sifting for insights is effort-intensive and time-consuming, highlighting the need for an efficient automated data exploration system to simplify the process.

**Existing Solutions.** To date, a number of data exploration systems have been proposed in the data management and data mining community (Bar El et al., 2020; Chanson et al., 2022; Personnaz et al., 2021; Cao et al., 2023). In general, these systems leverage a heuristic score function to identify the “best” data exploration sequence (a series of data analysis operations). While these systems show potential, they exhibit key limitations. **❶ User Intent Ignorance:** Existing tools are designed for general exploration and often fail to incorporate user intent. For instance, an analyst may be interested in understanding the economics-related factors but receive insights about demographics. **❷ Dataset Characteristic Ignorance:** They overlook dataset characteristics, often providing irrelevant insights. For instance, in a flight delay dataset, a correlation between flight delays and weather might yield insights but irrelevant factors like time and weather does not make sense in the context. They fall short in delivering a direct answers to the user question.

Recently, large language models (LLM) have shown promising potential in understanding user

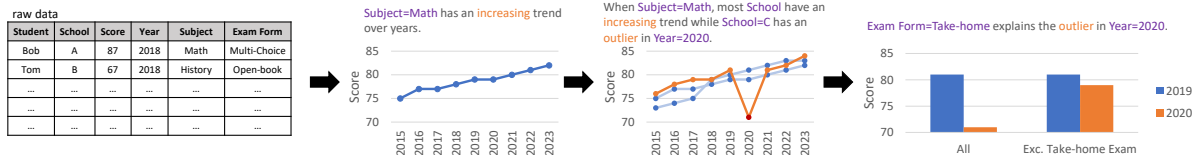


Figure 1: An example of data exploration.

intent and generating actions to achieve user-specified goals (Yao et al., 2022). In this regard, we anticipate that LLM can be leveraged to drive the data exploration process. However, there are several challenges that impede the adoption.

❸ **Hallucination:** Due to the infamous hallucination issue (Ji et al., 2023), LLMs often generate unreliable contents and are thus not mature for production use. ❹ **Overwhelming Context Window:** A dataset may contain millions of cells, which is overwhelming for LLMs to process.

**Our Solution.** To address these challenges, we propose *InsightPilot*, a system that automates data exploration using LLMs. This system facilitates exploration through the synergy of an LLM and an insight engine, which integrates three production-quality insight discovery tools: QuickInsight (Ding et al., 2019), MetaInsight (Ma et al., 2021), and XInsight (Ma et al., 2023) (detailed in Sec. 5.1). These tools offers a unified insight representation, enabling the LLM to engage coherently. The insight engine provides the LLM with accurate and reliable insights, avoiding hallucination. Furthermore, the insight engine presents a concise abstraction of the dataset to alleviate the overwhelming context window issue. In *InsightPilot*, users input high-level queries, like “show me the interesting trend in mathematics scores for students”. Then, the *InsightPilot* employ an LLM to interact with the insight engine using a set of carefully designed analysis actions to streamline common data exploration tasks. These actions serve as a coherent transition to chain up insights and generate a data exploration sequence to answer the user’s question. Finally, *InsightPilot* summarizes the results using natural language together with charts that are understandable to non-technical users.

**Contributions.** In summary, we make the following contributions: We propose *InsightPilot*, an automated system for data exploration that employs LLMs to drive the exploration process. *InsightPilot* streamlines the exploration process by interacting with an insight engine using a set of carefully designed analysis actions. We conduct a user study and a case study to demonstrate the effectiveness of *InsightPilot* in real-world scenarios.

## 2 Related Work

**Text-to-SQL.** To date, text-to-SQL is the most popular approach to enabling natural language interface to database. It translates users’ utterances into SQL queries for relational databases and has been studied by both database and NLP communities for several decades (Yu et al., 2018; Kim et al., 2020; Ma and Wang, 2022). Recent studies have shown that with LLM, text-to-SQL can now be augmented to support non-SQL enquiries such as entity extraction (Cheng et al., 2023). However, in EDA, users’ intents are often more complex than simple SQL queries. EDA generally involves more complicated user intents and goes beyond the expressiveness of basic SQL queries. We take a step further by using an LLM and analysis actions in *InsightPilot*, to produce natural and coherent data exploration sequences that accurately address users’ questions. This innovation provides an important complement to the existing text-to-SQL approach.

**Analytics Model in OLAP.** Traditionally, users interact with OLAP (online analytical processing) systems with a set of pre-defined operators (e.g., drill-down and roll-up) (Vassiliadis and Selis, 1999). Recently, there is a surge of interest in developing analytics models with higher-level abstraction and automation to facilitate complex OLAP needs (Vassiliadis et al., 2019). In *InsightPilot*, we use “analysis actions” to describe such high-level abstractions.

## 3 Preliminaries

In this section, we introduce the preliminaries of exploratory data analysis.

**Data Model.** Let  $D := \{X_1, \dots, X_n\}$  represents multi-dimensional data comprising  $n$  attributes, where each attribute  $X_i$  is either a dimension or a measure. A **dimension**  $X_i$  is a categorical attribute that can be used to group data. A **measure**  $X_i$  is a numerical attribute that can be used to perform aggregation operations. In *InsightPilot*, **filter** is the basic unit of data operations. Given a multi-dimensional data  $D$  and a dimension  $X$ , a filter  $p_i = X = x_i$  (e.g., “Subject=Math”) implies an equality assertion to  $X$  such that the value of  $X$

will equal  $x_i$ . A **subspace** is a conjunction of filters on disjoint dimensions (e.g., “Subject = Math AND Year = 2019”). A **breakdown** dimension is the dimension where the group-by operation is performed. Given a measure  $M$ , users may perform aggregation operations (such as SUM and AVG in SQL) over some records for  $M$ .

**Analysis Entity (AE).** An AE is defined as a 3-tuple  $AE := \langle agg(M), S, B \rangle$ , where  $M$  is a measure with an aggregation function  $agg$  applied,  $S$  is a subspace, and  $B$  is a breakdown dimension. It can be interpreted as an equivalent SQL query that performs aggregation operations over a set of records for the measure  $M$  in a subspace  $S$ , grouped by the breakdown dimension  $B$ . For instance, the AE  $\langle \text{AVG}(\text{Score}), \text{Subject} = \text{Math}, \text{Year} \rangle$  is equivalent to the SQL query `SELECT AVG(Score) FROM Table WHERE Subject = Math GROUP BY Year`.

**Data Insight.** A basic data insight is represented as a 3-tuple  $\langle AE, Type, Property \rangle$ . Here,  $AE$  denotes an analysis entity,  $Type$  specifies the insight’s kind (e.g., trend, outlier), and  $Property$  encapsulates additional outputs from insight mining algorithms, like extreme points for a unimodality pattern. We categorize insights into **basic insights**, directly derived from data (e.g., trend insights), and **compound insights**, which build upon other insights. A *meta-insight*, for instance, summarizes several similar insights (e.g., sales trends across various cities). Both insight categories can be expressed as the 3-tuple format. Throughout this paper, “insight” pertains to both types. We have crafted templates to articulate these insights in user-friendly language, accompanied by visualizations.

## 4 Problem Definition

In this section, we define the problem of generating a sequence of data insights to address users’ analysis intents.

**Analysis Actions.** An analysis action is defined as a transition from one data insight to several other data insights (which may also be no insight or solely one insight). In our context, an action represents a reasonable data analysis operation with the existing knowledge (e.g., user question, dataset, and explored insights). This is defined as a function  $AA : \text{Insight} \rightarrow \text{Insight}^*$ , where  $\text{Insight}^*$  denotes the set of all possible data insights. In addition, we define two special analysis actions, namely,  $AA_{init}$  and  $AA_{back}$ .  $AA_{init}$  that takes a dataset as input and provides a set of initial insights and  $AA_{back}$  that

backtracks to the last state and returns the insights generated by the preceding analysis action.

**Data Exploration Sequence.** A data exploration sequence is defined as a series of data insights interconnected by analysis actions. It is represented as  $\mathcal{S} = \langle AA_{init}, \text{Insight}_1, AA_1, \dots, AA_n, \perp \rangle$ , where  $\text{Insight}_i$  is a data insight picked from the output of the preceding analysis action  $AA_{i-1}$ ,  $AA_i$  is an analysis action, and  $\perp$  symbolizes the termination. Each analysis action  $AA_i$  takes the preceding data insight  $\text{Insight}_{i-1}$  as input and produces multiple insights to be picked for the next analysis action.

**Generating Final Answer.** Given a user question  $Q$  and the data exploration sequence  $\mathcal{S}$ , we define the problem of generating a final answer as a function  $FA : Q \times \text{flatten}(\mathcal{S}) \rightarrow A$ , where  $A$  is the final answer to the user question  $Q$  and  $\text{flatten}(\mathcal{S})$  is the top-k of all data insights generated in the data exploration sequence (including unpicked insights). The final answer  $A$  can be obtained by any document QA techniques over  $\text{flatten}(\mathcal{S})$ .

**Application Scope.** In *InsightPilot*, we focus on the class of data analysis tasks are expressed as *fuzzy* and *high-level* tasks. These are tasks where the user’s intent is not explicitly clear or the analysis objective is often complex, requiring multiple steps to fully address. For example, a fuzzy high-level task could be “Analyze sales performance over the past year.” The exact steps required to answer this question are not specified, and a variety of different analysis actions and insights may be required to provide a comprehensive answer. Note that while we focus on this specific class of questions, the unique feature of *InsightPilot* can be integrated with other systems (e.g., text-to-SQL) and support diverse data analysis scenarios.

## 5 InsightPilot Design

Figure 2 depicts the overview of *InsightPilot*. Overall, *InsightPilot* constitutes a pipeline of three components: (1) a user interface that enables users to issue inquiries in natural language, and also depicts analysis results in texts and charts; (2) a LLM that drives the exploration process by selecting appropriate insights and analysis intents based on the context (e.g., user question, dataset domain knowledge, and current exploration state); (3) an insight engine that executes analysis action, generates insights, and presents results in natural language.

**Working Example.** In Figure 2, *InsightPilot* is illustrated using the example from Figure 1. A user



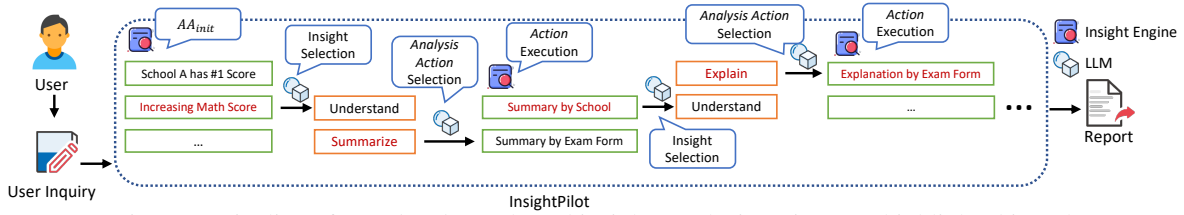


Figure 2: Pipeline of *InsightPilot*. Selected insights/analysis actions are highlighted in red.

poses a query: “*show me any interesting trend in mathematics scores for students*”. The insight engine then generates initial insights with  $AA_{init}$ . One such insight might be “*School A has the Rank#1 average score.*” Based on the user’s question, the LLM identifies the most pertinent insight, such as “*the mathematics scores of students have been increasing over time*”, using predefined prompts (refer Sec. 5.2). After choosing the insight, we prepare potential analysis actions and the LLM selects an appropriate one, in this case, *compare* (details in Sec. 5.1). Executing this action, the insight engine summarizes the math scores trend across schools. It observes: “*most schools show rising math scores, except for an outlier in 2020 for school C.*” To delve deeper, the LLM continues to select insights and actions, eventually querying the engine to “*explain the 2020 outlier for school C*”.

Interactions continue until the LLM completes its exploration (i.e., choose  $\perp$  as the next action) or hits the token size limit. Once done, insights are translated to natural language for the prompt. Given the typically large number of insights, insight ranking becomes crucial. The insight engine then presents the top- $K$  insights, which the LLM condenses into a coherent report. This report and the top- $K$  insights (in the form of charts) are then displayed to the user via the interface.

## 5.1 Analysis Action

Every time the LLM selects an insight and an analysis action, the insight engine will execute the action and generate new insights. Currently, we have prepared four analysis actions for the LLM to select: *understand*, *summarize*, *compare*, and *explain*. These actions are in accordance with three insight discovery solutions, namely QuickInsight (Ding et al., 2019) for *understand*, MetaInsight (Ma et al., 2021) for *summarize* and *compare*, and XInsight (Ma et al., 2023) for *explain*. We now elaborate on the design of these analysis actions.

① **Understand.** This action is designed to help users understand the high-level patterns in the data. In particular, it attempts to enumerate all possi-

ble AEs (see definition in Sec. 3) under the AE of input insight, applies the insight mining algorithm (e.g., trend detection) on each AE to identify basic insight and transforms them into human-understandable natural language.

② **Summarize.** This action aims to view an input insight from various angles. Starting with a basic insight (like a trend), it employs a specific insight mining algorithm on the AEs of the input to verify the presence of the primary insight type and property (e.g., an increasing trend) across each AE. If consistent across all AEs, the output is “*the basic insight type and property are universal among AEs*”. If not, it’s “*the basic insight type and property are present in some AEs*”. Using the example insight of an “*increasing math score trend*”, the outcome could be “*most schools show rising math scores, barring school C*” or “*in most subjects, scores have risen over time*”. These compound insights can be further explored by subsequent analysis actions.

③ **Compare.** This action shares a similar design with *summarize*. It is designed to compare the input insight from different neighbors. In particular, it starts with a basic insight (e.g., a trend) and then applies the particular insight mining algorithm on the neighboring AEs. Given an input insight “*the increasing trend of the mathematics scores in school A*”, it will generate a comparison of the trend regarding different schools, e.g., “*the mathematics scores of students in school A and B have been increasing over time, except school C.*” This comparison is also represented by a compound insight.

④ **Explain.** This action is designed to explain the insight that reveals a difference or an outlier in the data. It supports both basic insights (e.g., a outlier insight or a change point insight) and compound insights (e.g., a summary insight with an exceptional case). Given the input insight with difference or outlier, it will identify a subspace that is responsible for the outcome using causal inference and then constitute a new compound insight to encode the cause. For example, given “*the outlier of the mathematics scores in 2020 for school C*”, it will identify explanations such as “*the difference on the mathe-*

mathematics scores of students in school C between 2019 and 2020 is caused by Exam Form=Take-home. When excluding Exam Form=Take-home, 2020 is no longer an outlier.”

## 5.2 Prompt Engineering

To deliver a self-contained presentation, we describe the design of our prompt engineering techniques. In general, any agent-based prompt template (e.g., ReACT (Yao et al., 2022)) can be used to instantiate *InsightPilot*. We prepare separate prompt templates for different stages of *InsightPilot* for selecting insights and analysis actions, and for finalizing the answer. Routine instructions are used in the prompt to improve the usefulness, clarity, and coherence of the LLM outputs.

## 5.3 Insight Ranking

Consider the final answer generation phase detailed in Sec. 4. The insight engine often yields an overwhelming number of insights, sometimes reaching hundreds within a single exploration sequence. Given the LLM’s capacity, it is infeasible to process all these insights. Thus, we prioritize by extracting the top- $K$  insights. Notably, the value of  $K$  surpasses the count of selected insights in the exploration sequence ( $Insight \in \mathcal{S}$ ), ensuring the chosen insights are encompassed within the top- $K$ . Next, we present three schemes to rank top- $K$  insights.

**Redundant Insight Elimination.** Trivial insights can be eliminated if they are entailed by a more informative insight. For example, if three insight #1: “School=A has the highest mathematics score,” #2: “School=A has the highest score in 2022,” and #3 “School=A has the highest mathematics score in 2022” are generated, we can exclude the third insight since it trivially derives from the first two. Enlightened by this observation, we propose to eliminate insights that are entailed by other insights. In particular, such elimination is achieved by identifying insights by looping through every possible pair of insights and checking if there exists a dominator to make one of them trivial.

**Semantic Similarity-based Elimination.** To further decrease the number of insights for the LLM to handle, we employ semantic similarity. We identify the top- $K'$  ( $K' \gg K$ ) insights most relevant to the user’s question using an embedding model that transforms text inputs into vector representations. By calculating the cosine similarity between each insight’s vector and the user’s question, we rank the insights, selecting the top- $K'$  most relevant ones.

This method not only reduces the LLM’s processing load, but it also ensures the retained insights align closely with the user’s question.

**Diversity-aware Reranking.** After applying the above two strategies, we obtain the top- $K'$  insights. Then, we seek to re-rank them according to their diversity to provide the top- $K$  insight. In the context of recommending a set of insights, the goal is to select insights with high individual scores and low redundancy. This can be thought of as maximizing the total usefulness of the selected insights. To achieve this, we leverage the second-order approximated ranking algorithm as explained in (Ma et al., 2021) to determine the order.

## 6 Evaluation

**Implementation.** We implement our tool based on the codebases of QuickInsight, MetaInsight, and XInsight, adding an additional 1.8K lines of C# code and 1.5K lines of JavaScript code. We use “gpt-3.5-turbo” as our language model and “text-embedding-ada-002” is used to generate embeddings. Both models are provided by OpenAI.

|                          | <i>InsightPilot</i> | Code Interpreter | Pandas Agent |
|--------------------------|---------------------|------------------|--------------|
| <b>Relevance</b>         | <b>4.50±0.76</b>    | 4.08±0.86        | 1.92±1.00    |
| <b>Completeness</b>      | <b>4.67±0.55</b>    | 3.54±1.00        | 1.12±0.33    |
| <b>Understandability</b> | <b>4.46±0.64</b>    | 4.25±0.83        | 1.62±0.90    |

Table 1: Results of the User Study

**User Study.** We conduct a user study to simulate the real-world application of *InsightPilot*, highlighting its unique advantages over existing solutions like OpenAI Code Interpreter (OpenAI, 2023) and Langchain Pandas Agent (Langchain, 2023), both being *state-of-the-art* in their domains. We explored but excluded text-to-SQL models like FlanT5 (Chung et al., 2022) and Anthropic Claude (Anthropic, 2023), due to their inability to provide direct answers or load the datasets. Four independent data science participants are recruited for the study. They are given two datasets and asked to raise three questions each within the *InsightPilot* application scope, resulting in 24 groups of comparisons (4 participants  $\times$  2 datasets  $\times$  3 questions). They score the systems on *Relevance*, *Completeness* and *Understandability* (scale of 1 to 5).

Results are reported in Table 1. *InsightPilot* consistently outperforms the others in all three metrics, showcasing its capability in offering relevant, complete and understandable responses. Specifically, *InsightPilot* is notably better than the Code

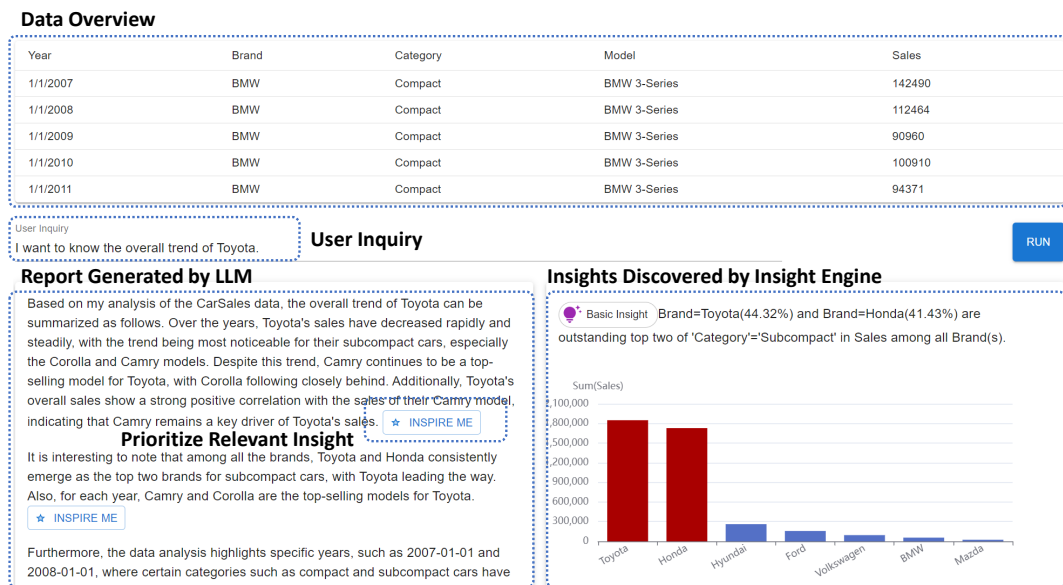


Figure 3: User interface of *InsightPilot*.

Interpreter and Pandas Agent in *completeness* ( $p$ -value  $< 0.05$ ). Upon examining the competitors' responses, we found they often provide an ad-hoc answer to a specific region of the dataset. For example, when inquiring about differences in car sales between Mazda and Toyota, competitors reveal only the overall difference, whereas *InsightPilot* further analyzes various breakdowns, identifying that "Toyota's Corolla model accounts for a larger percentage of sales compared to Mazda's models." **Case Study.** We demonstrate *InsightPilot*'s use in Figure 3, showcasing a portion of its output (due to space limit). The user is using a car sales dataset and enquiries "I want to know the overall trend of Toyota." *InsightPilot* first identifies that "Toyota has a decreasing trend on its sales over the years" and then dives into two representative models of Toyota, namely "Toyota Corolla" and "Toyota Camry." It identifies that "Corolla" and "Camry" constitute two top-selling models of Toyota and the sales of "Camry" has a strong correlation with the overall Sales of Toyota. Therefore, *InsightPilot* concludes that the Camry is the key driver of Toyota's sales. Afterwards, *InsightPilot* further compares Toyota with Honda and identifies that they are the top-two brands for subcompact cars while Toyota leading the way. Besides, *InsightPilot* also looks into the sales of Toyota in different years and obtains other interesting insights.

## 7 Discussion

**Action-wise Performance.** The efficacy of *InsightPilot*, an automated data analytics tool, hinges on

the comprehensive design of each action and its accurate execution. While we introduce four actions rooted in common data analysis techniques, it is vital to note that *InsightPilot*'s innovation is not tied to specific action designs. Instead, its uniqueness lies in leveraging LLM for data exploration.

**Comprehensive Assessment.** To validate *InsightPilot*'s effectiveness, it is essential to evaluate it across diverse real-life datasets and dimensions, such as keyword preservation. Nonetheless, *InsightPilot* often produces open-ended responses, making manual evaluation crucial for assessing answer quality. These hurdles make it challenging to efficiently evaluate *InsightPilot*'s performance in a comprehensive manner. An LLM-based evaluation framework could potentially streamline this process (Wang et al., 2023; Li et al., 2023).

## 8 Conclusion

In this paper, we introduce *InsightPilot*, an LLM-empowered automated data exploration system. By seamlessly integrating LLM with state-of-the-art insight engines, *InsightPilot* streamlines data analysis into a coherent exploration sequence. Effective for real-world datasets, it allows users to derive insights via natural language inquiries. *InsightPilot* equips even non-technical individuals to benefit from data analysis, bolstering efficiency and data-driven decision-making.

## Acknowledgement

Rui Ding and Shuai Wang are the corresponding authors.

## References

- Anthropic. 2023. Anthropic claude chat. <https://claude.ai/chat/>.
- Ori Bar El, Tova Milo, and Amit Somech. 2020. Automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1527–1537.
- Yukun Cao, Xike Xie, and Kexin Huang. 2023. Learn to explore: on bootstrapping interactive data exploration with meta-learning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1720–1733. IEEE.
- Alexandre Chanson, Nicolas Labroche, Patrick Marcel, Stefano Rizzi, and Vincent t’Kindt. 2022. Automatic generation of comparison notebooks for interactive data exploration. In *EDBT*, pages 2–274.
- Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2023. Binding language models in symbolic languages. *International Conference on Learning Representations*.
- 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*.
- Jay Devore. 2007. Making sense of data: A practical guide to exploratory data analysis and data mining.
- Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. Quickinsights: Quick and automatic discovery of insights from multi-dimensional data. In *ACM SIGMOD International Conference on Management of Data*.
- Andrew T Jebb, Scott Parrigon, and Sang Eun Woo. 2017. Exploratory data analysis as a foundation of inductive research. *Human Resource Management Review*, 27(2):265–276.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.
- Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to sql: Where are we today? *VLDB*.
- Matthieu Komorowski, Dominic C. Marshall, Justin D. Saliccioli, and Yves Crutain. 2016. *Exploratory Data Analysis*, pages 185–203. Springer International Publishing, Cham.
- Langchain. 2023. Langchain pandas dataframe agent. <https://python.langchain.com/docs/integrations/toolkits/pandas>.
- Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Tianxiang Li, Shuai Wang, Cuiyun Gao, and Yang Liu. 2023. Split and merge: Aligning position biases in large language model based evaluators. *arXiv preprint arXiv:2310.01432*.
- Pingchuan Ma, Rui Ding, Shi Han, and Dongmei Zhang. 2021. Metainsight: Automatic discovery of structured knowledge for exploratory data analysis. In *ACM SIGMOD International Conference on Management of Data*.
- Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. Xinsight: explainable data analysis through the lens of causality. In *ACM SIGMOD International Conference on Management of Data*.
- Pingchuan Ma and Shuai Wang. 2022. Mt-teql: Evaluating and augmenting neural nllm on real-world linguistic and schema variations. *VLDB*.
- OpenAI. 2023. Openai code interpreter. <https://chat.openai.com/?model=gpt-4-code-interpreter>.
- Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. 2021. Balancing familiarity and curiosity in data exploration with deep reinforcement learning. In *Fourth Workshop in Exploiting AI Techniques for Data Management*, pages 16–23.
- Panos Vassiliadis, Patrick Marcel, and Stefano Rizzi. 2019. Beyond roll-up’s and drill-down’s: An intentional analytics model to reinvent olap. *Information Systems*, 85:68–91.
- Panos Vassiliadis and Timos Sellis. 1999. A survey of logical models for olap databases. *ACM Sigmod Record*, 28(4):64–69.
- Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

# SynJax: Structured Probability Distributions for JAX

Miloš Stanojević  
Google DeepMind  
stanojevic@google.com

Laurent Sartran  
Google DeepMind  
lsartran@google.com

## Abstract

The development of deep learning software libraries enabled significant progress in the field by allowing users to focus on modeling, while letting the library to take care of the tedious and time-consuming task of optimizing execution for modern hardware accelerators. However, this has benefited only particular types of deep learning models, such as Transformers, whose primitives map easily to the vectorized computation. The models that explicitly account for structured objects, such as trees and segmentations, did not benefit equally because they require custom algorithms that are difficult to implement in a vectorized form.

SynJax directly addresses this problem by providing an efficient vectorized implementation of inference algorithms for structured distributions covering alignment, tagging, segmentation, constituency trees and spanning trees. This is done by exploiting the connection between algorithms for automatic differentiation and probabilistic inference. With SynJax we can build large-scale differentiable models that explicitly model structure in the data. The code is available at <https://github.com/google-deepmind/synjax>.

## 1 Introduction

In many domains, data can be seen as having some structure explaining how its parts fit into a larger whole. This structure is often latent, and it varies depending on the task. For examples of discrete structures in natural language consider Figure 1. The words together form a sequence. Each word in a sequence is assigned a part-of-speech tag. These tags are dependent on each other, forming a linear-chain marked in red. The words in the sentence can be grouped together into small disjoint contiguous groups by sentence segmentation, shown with bubbles. A deeper analysis of language would show that the groupings can be done recursively and thereby produce a syntactic tree structure. Structures can also relate two languages. For instance,

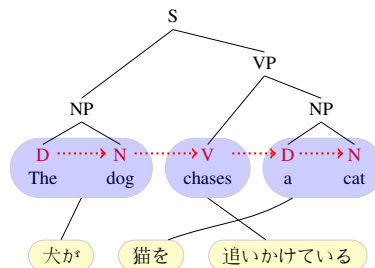


Figure 1: Examples of natural language structures.

in the same figure, a Japanese translation can be mapped to an English source by an alignment.

These structures are not specific to language. Similar structures appear in biology as well. Nucleotides of any two RNA sequences are matched with monotone alignment (Needleman and Wunsch, 1970; Wang and Xu, 2011), genomic data is segmented into contiguous groups (Day et al., 2007) and tree-based models of RNA capture the hierarchical nature of the protein folding process (Sakakibara et al., 1994; Hockenmaier et al., 2007; Huang et al., 2019).

Most contemporary deep learning models attempt to predict output variables directly from the input without any explicit modeling of the intermediate structure. Modeling structure explicitly could improve these models in multiple ways. First, it could allow for better generalization through the right inductive biases (Dyer et al., 2016; Sartran et al., 2022). This would improve not only sample efficiency but also downstream performance (Bastings et al., 2017; Nădejde et al., 2017; Bisk and Tran, 2018). Explicit modeling of structure can also enable incorporation of problem specific algorithms (e.g. finding shortest paths; Pogančić et al., 2020; Niepert et al., 2021) or constraints (e.g. enforcing alignment Mena et al., 2018 or enforcing compositional calculation Havrylov et al., 2019). Discrete structure also allows for better interpretability of the model’s decisions (Bastings

et al., 2019). Finally, sometimes structure is the end goal of learning itself – for example we may know that there is a hidden structure of a particular form explaining the data, but its specifics are not known and need to be discovered (Kim et al., 2019; Paulus et al., 2020).

Auto-regressive models are the main approach used for modeling sequences. Non-sequential structures are sometimes linearized and approximated with a sequential structure (Choe and Charniak, 2016). These models are powerful as they do not make any independence assumptions and can be trained on large amounts of data. While sampling from auto-regressive models is typically tractable, other common inference problems like finding the optimal structure or marginalizing over hidden variables are not tractable. Approximately solving these tasks with auto-regressive models requires using biased or high-variance approximations that are often computationally expensive, making them difficult to deploy in large-scale models.

Alternative to auto-regressive models are models over factor graphs that factorize in the same way as the target structure. These models can efficiently compute all inference problems of interest exactly by using specialized algorithms. Despite the fact that each structure needs a different algorithm, we do not need a specialized algorithm for each inference task (argmax, sampling, marginals, entropy etc.). As we will show later, SynJax uses automatic differentiation to derive many quantities from just a single function per structure type.

Large-scale deep learning has been enabled by easy to use libraries that run on hardware accelerators. Research into structured distributions for deep learning has been held back by the lack of ergonomic libraries that would provide accelerator-friendly implementations of structure components – especially since these components depend on algorithms that often do not map directly onto available deep learning primitives, unlike Transformer models. This is the problem that SynJax addresses by providing easy to use structure primitives that compose within JAX machine learning framework.

To see how easy it is to use SynJax consider example in Figure 2. This code implements a policy gradient loss that requires computing multiple quantities – sampling, argmax, entropy, log-probability – each requiring a different algorithm. In this concrete code snippet, the structure is a non-projective directed spanning tree with a single root

```
@typed
def policy_gradient_loss(
    log_potentials: Float[jax.Array, "*batch_n_n"],
    key: jax.random.KeyArray) -> Float[jax.Array, ""]:
    dist = synjax.SpanningTreeCRF(log_potentials,
        directed=True, projective=False, single_root_edge=True)
    # Sample from policy
    sample = dist.sample(key)
    # Get reward
    reward = reward_fn(sample)
    # Compute log-prob
    log_prob = dist.log_prob(sample)
    # Self-critical baseline
    baseline = reward_fn(dist.argmax())
    # REINFORCE
    objective = stop_gradient(reward-baseline) * log_prob
    # Entropy regularization
    return -jnp.mean(objective + 0.5*dist.entropy())
```

Figure 2: Example of implementing policy gradient with self-critical baseline and entropy regularization for spanning trees.

edge constraint. Because of that SynJax will:

- compute argmax with Tarjan’s (1977) maximum spanning tree algorithm adapted for single root edge trees (Stanojević and Cohen, 2021),
- sample with Wilson’s (1996) sampling algorithm for single root trees (Stanojević, 2022),
- compute entropy with Matrix-Tree Theorem (Tutte, 1984) adapted for single root edge trees (Koo et al., 2007; Zmigrod et al., 2021).

If the user wants only to change slightly the tree requirements to follow the *projectivity constraint* they only need to change one flag and SynJax will in the background use completely different algorithms that are appropriate for that structure: it will use Kuhlmann’s algorithm (2011) for argmax and variations of Eisner’s (1996) algorithm for other quantities. The user does not need to implement any of those algorithms or even be aware of their specifics, and can focus on the modeling side of the problem.

## 2 Structured Distributions

Distributions over most structures can be expressed with factor graphs – bipartite graphs that have random variables and factors between them. We associate to each factor a non-negative scalar, called potential, for each possible assignment of the random variables that are in its neighbourhood. The potential of the structure is a product of its factors:

$$\phi(t) = \prod_{e \in t} \phi(e) \quad (1)$$

where  $t$  is a structure,  $e$  is a factor/part, and  $\phi(\cdot)$  is the potential function. The probability of a struc-

ture can be found by normalizing its potential:

$$p(t) = \frac{\prod_{e \in t} \phi(e)}{\sum_{t' \in T} \prod_{e' \in t'} \phi(e')} = \frac{\phi(t)}{Z} \quad (2)$$

where  $T$  is the set of all possible structures and  $Z$  is a normalization often called partition function. This equation can be thought of as a *softmax* equivalent over an extremely large set of structured outputs that share sub-structures (Sutton and McCallum, 2007; Mihaylova et al., 2020).

### 3 Computing Probability of a Structure and Partition Function

Equation 2 shows the definition of the probability of a structure in a factor graph. Computing the numerator is often trivial. However, computing the denominator, the partition function, is the complicated and computationally demanding part because the set of valid structures  $T$  is usually exponentially large and require specialized algorithms for each type of structure. As we will see later, the algorithm for implementing the partition function accounts for the majority of the code needed to add support for a structured distribution, as most of the other properties can be derived from it. Here we document the algorithms for each structure.

#### 3.1 Sequence Tagging

Sequence tagging can be modelled with Linear-Chain CRF (Lafferty et al., 2001). The partition function for linear-chain models is computed with the forward algorithm (Rabiner, 1990). The computational complexity is  $\mathcal{O}(m^2n)$  for  $m$  tags and sequence of length  $n$ . Särkkä and García-Fernández (2021) have proposed a parallel version of this algorithm that has parallel computational complexity  $\mathcal{O}(m^3 \log n)$  which is efficient for  $m \ll n$ . Rush (2020) reports a speedup using this parallel method for Torch-Struct, however in our case the original forward algorithm gave better performance both in terms of speed and memory.

The SynJax implementation of Linear-Chain CRF supports having a different transition matrix for each time step which gives greater flexibility needed for implementing models like LSTM-CNN-CRF (Ma and Hovy, 2016) and Neural Hidden Markov Model (Tran et al., 2016).

#### 3.2 Segmentation with Semi-Markov CRF

Joint segmentation and tagging can be done with a generalization of linear-chain called Semi-Markov

CRF (Sarawagi and Cohen, 2004; Abdel-Hamid et al., 2013; Lu et al., 2016). It has a similar parametrization with transition matrices except that here transitions can jump over multiple tokens. The partition function is computed with an adjusted version of the forward algorithm that runs in  $\mathcal{O}(sm^2n)$  where  $s$  is the maximal size of a segment.

### 3.3 Alignment Distributions

Alignment distributions are used in time series analysis (Cuturi and Blondel, 2017), RNA sequence alignment (Wang and Xu, 2011), semantic parsing (Lyu and Titov, 2018) and many other areas.

#### 3.3.1 Monotone Alignment

Monotone alignment between two sequences of lengths  $n$  and  $m$  allows for a tractable partition function that can be computed in  $\mathcal{O}(nm)$  time using the Needleman-Wunsch (1970) algorithm.

#### 3.3.2 CTC

Connectionist Temporal Classification (CTC, Graves et al., 2006; Hannun, 2017) is a monotone alignment model widely used for speech recognition and non-auto-regressive machine translation models. It is distinct from the standard monotone alignment because it requires special treatment of the *blank symbol* that provides jumps in the alignment table. It is implemented with an adjusted version of Needleman-Wunsch algorithm.

#### 3.3.3 Non-Monotone 1-on-1 Alignment

This is a bijective alignment that directly maps elements between two sets given their matching score. Computing partition function for this distribution is intractable (Valiant, 1979), but we can compute some other useful quantities (see Section 5).

### 3.4 Constituency Trees

#### 3.4.1 Tree-CRF

Today’s most popular constituency parser by Kitaev et al. (2019) uses a global model with factors defined over labelled spans. Stern et al. (2017) have shown that inference in this model can be done efficiently with a custom version of the CKY algorithm in  $\mathcal{O}(mn^2 + n^3)$  where  $m$  is number of non-terminals and  $n$  is the sentence length.

#### 3.4.2 PCFG

Probabilistic Context-Free Grammars (PCFG) are a generative model over constituency trees where each grammar rule is associated with a locally normalized probability. These rules serve as a template

which, when it gets expanded, generates jointly a constituency tree together with words as leaves.

SynJax computes the partition function using a vectorized form of the CKY algorithm that runs in cubic time. Computing a probability of a tree is in principle simple: just enumerate the rules of the tree, look up their probability in the grammar and multiply the found probabilities. However, extracting rules from the set of labelled spans requires many sparse operations that are non-trivial to vectorize. We use an alternative approach where we use *sticky* span log-potentials to serve as a mask for each constituent: constituents that are part of the tree have sticky log-potentials 0 while those that are not are  $-\infty$ . With sticky log-potentials set in this way computing log-partition provides a log-probability of a tree of interest.

### 3.4.3 TD-PCFG

Tensor-Decomposition PCFG (TD-PCFG, [Cohen et al., 2013](#); [Yang et al., 2022](#)) uses a lower rank tensor approximation of PCFG that makes inference with much larger number of non-terminals feasible.

## 3.5 Spanning Trees

Spanning trees appear in the literature in many different forms and definitions. We take a spanning tree to be any subgraph that connects all nodes and does not have cycles. We divide spanning tree CRF distributions by the following three properties:

**directed or undirected** Undirected spanning trees are defined over symmetric weighted adjacency matrices i.e. over undirected graphs. Directed spanning trees are defined over directed graphs with special root node.

**projective or non-projective** Projectivity is a constraint that appears often in NLP. It constrains the spanning tree over words not to have crossing edges. Non-projective spanning tree is just a regular spanning tree – i.e. it may not satisfy the projectivity constraint.

**single root edge or multi root edges** NLP applications usually require that there can be only one edge coming out of the root ([Zmigrod et al., 2020](#)). Single root edge spanning trees satisfy that constraint.

Each of these choices has direct consequences on which algorithm should be used for probabilistic inference. SynJax abstracts away this from the user and offers a unified interface where the user only

needs to provide the weighted adjacency matrix and set the three mentioned boolean values. Given the three booleans SynJax can pick the correct and most optimal algorithm. In total, these parameters define distributions over 8 different types of spanning tree structures all unified in the same interface. We are not aware of any other library providing this set of unified features for spanning trees.

We reduce undirected case to the rooted directed case due to bijection. For projective rooted directed spanning trees we use Eisner’s algorithm for computation of the partition function ([Eisner, 1996](#)). The partition function of Non-Projective spanning trees is computed using Matrix-Tree Theorem ([Tutte, 1984](#); [Koo et al., 2007](#); [Smith and Smith, 2007](#)).

## 4 Computing Marginals

In many cases we would like to know the probability of a particular part of structure appearing, regardless of the structure that contains it. In other words, we want to marginalize (i.e. sum) the probability of all the structures that contain that part:

$$p(e) = \sum_{t \in T} \mathbb{1}[e \in t] p(t) = \sum_{t' \in T_e} p(t') \quad (3)$$

where  $\mathbb{1}[\cdot]$  is the indicator function,  $T$  is the set of all structures and  $T_e$  is the set of structures that contain factor/part  $e$ .

Computing these factors was usually done using specialized algorithms such as Inside-Outside or Forward-Backward. However, those solutions do not work on distributions that cannot use belief propagation like Non-Projective Spanning Trees. A more general solution is to use an identity that relates gradients of factor’s potentials with respect to the log-partition function:

$$p(e) = \frac{\partial \log Z}{\partial \phi(e)} \quad (4)$$

This means that we can use any differentiable implementation of log-partition function as a forward pass and apply backpropagation to compute the marginal probability ([Darwiche, 2003](#)). [Eisner \(2016\)](#) has made an explicit connection that “Inside-Outside and Forward-Backward algorithms are just backprop”. This approach also works for Non-Projective Spanning Trees that do not fit belief propagation framework ([Zmigrod et al., 2021](#)).

For template models like PCFG, we use again the *sticky* log-potentials because usually we are



not interested in marginal probability of the rules but in the marginal probability of the instantiated constituents. The derivative of log-partition with respect to the constituent’s *sticky* log-potential will give us marginal probability of that constituent.

## 5 Computing Most Probable Structure

For finding the score of the highest scoring structure we can just run the same belief propagation algorithm for log-partition, but with the *max-plus semiring* instead of the log-plus semiring (Goodman, 1999). To get the most probable structure, and not just its potential, we can compute the gradient of part potentials with respect to the viterbi structure potential (Rush, 2020).

The only exceptions to this process are non-monotone alignments and spanning trees because they do not fit easily in belief propagation framework. For the highest scoring non-monotone alignment, we use the Jonker–Volgenant algorithm as implemented in SciPy (Crouse, 2016; Virtanen et al., 2020). Maximal *projective* spanning tree can be found by combining Eisner’s algorithm with max-plus semiring, but we have found Kuhlmann’s tabulated arc-hybrid algorithm to be much faster (Kuhlmann et al., 2011) (see Figure 4 in the appendix). This algorithm cannot be used for any inference task other than argmax because it allows for spurious derivations. To enforce single-root constraint with Kuhlmann’s algorithm we use the Reweighting trick from Stanojević and Cohen (2021). For *non-projective* spanning trees SynJax uses a combination of Reweighting+Tarjan algorithm as proposed in Stanojević and Cohen (2021).

## 6 Sampling a Structure

Strictly speaking, there is no proper sampling semiring because semirings cannot have non-deterministic output. However, we can still use the semiring framework and make some aspect of them non-deterministic. Aziz (2015) and Rush (2020) use a semiring that in the forward pass behaves like a log-semiring, but in the backward pass instead of computing the gradient it does sampling. This is in line of how forward-filtering backward-sampling algorithm works (Murphy, 2012, §17.4.5).

Non-Projective Spanning Trees do not support the semiring framework so we use custom algorithms for them described in Stanojević (2022). It contains Colbourn’s algorithm that has a fixed runtime of  $\mathcal{O}(n^3)$  but is prone to numerical issues be-

cause it requires matrix-inversion (Colbourn et al., 1996), and Wilson’s algorithm that is more numerically stable but has a runtime that depends on concrete values of log-potentials (Wilson, 1996). SynJax also supports vectorized sampling without replacement (SWOR) from Stanojević (2022).

## 7 Differentiable Sampling

The mentioned sampling algorithms provide unbiased samples of structures useful for many inference tasks, but they are not differentiable because the gradient of sampling from discrete distributions is zero almost everywhere. This problem can be addressed with log-derivative trick from REINFORCE algorithm (Williams, 1992), but that provides high variance estimates of gradients. To address this problem there have been different proposals for differentiable sampling algorithms that are biased but can provide low-variance estimates of gradients. SynJax implements majority of the main approaches in the literature including structured attention (Kim et al., 2017), relaxed dynamic programming (Mensch and Blondel, 2018), Perturb-and-MAP (Corro and Titov, 2019), Gumbel-CRF (Fu et al., 2020), Stochastic Softmax-Tricks (Paulus et al., 2020), and Implicit Maximum-Likelihood estimation (Niepert et al., 2021). It also includes different noise distributions for perturbations models, including Sum-of-Gamma noise (Niepert et al., 2021) that is particularly suited for structured distributions.

## 8 Entropy and KL Divergence

To compute the cross-entropy and KL divergence, we will assume that the two distributions factorize in exactly the same way. Like some other properties, cross-entropy can also be computed with the appropriate semirings (Hwa, 2000; Eisner, 2002; Cortes et al., 2008; Chang et al., 2023), but those approaches would not work on Non-Projective Spanning Tree distributions. There is a surprisingly simple solution that works across all distributions that factorize in the same way and has appeared in a couple of works in the past (Li and Eisner, 2009; Martins et al., 2010; Zmigrod et al., 2021). Here

we give a full derivation for cross-entropy:

$$\begin{aligned}
H(p, q) &= - \sum_{t \in T} p(t) \log q(t) \\
&= \log Z_q - \sum_{t \in T} p(t) \sum_{e \in t} \log \phi_q(e) \\
&= \log Z_q - \sum_{t \in T} p(t) \sum_{e \in E} \mathbb{1}[e \in t] \log \phi_q(e) \\
&= \log Z_q - \sum_{e \in E} p(e) \log \phi_q(e) \quad (5)
\end{aligned}$$

This reduces the computation of cross-entropy to finding marginal probabilities of one distribution, and finding log-partition of the other – both of which can be computed efficiently for all distributions in SynJax. Given the method for computing cross-entropy, finding entropy is trivial:

$$H(p) = H(p, p) \quad (6)$$

KL divergence is easy to compute too:

$$D_{KL}(p||q) = H(p, q) - H(p) \quad (7)$$

## 9 Library Design

Each distribution has different complex shape constraints which makes it complicated to document and implement all the checks that verify that the user has provided the right arguments. The *jaxtyping* library<sup>1</sup> was very valuable in making SynJax code concise, documented and automatically checked.

Structured algorithms require complex broadcasting, reshaping operations and application of semirings. To make this code simple, we took the *einsum* implementation from the core JAX code and modified it to support arbitrary semirings. This made the code shorter and easier to read.

Most inference algorithms apply a large number of elementwise and reshaping operations that are in general fast but create a large number of intermediate tensors that occupy memory. To speed this up we use checkpointing (Griewank, 1992) to avoid memorization of tensors that can be recomputed quickly. That has improved memory usage *and* speed, especially on TPUs.

All functions that could be vectorized are written in pure JAX. Those that cannot, like Wilson sampling (1996) and Tarjan’s algorithm (1977), are implemented with Numba (Lam et al., 2015).

<sup>1</sup><https://github.com/google/jaxtyping>

| Distribution       | Torch-Struct | SynJax |              | Speedup |
|--------------------|--------------|--------|--------------|---------|
|                    | LoC          | LoC    | (relative %) |         |
| Linear-Chain-CRF   | 32           | 15     | (46%)        | 13×     |
| Semi-Markov CRF    | 54           | 15     | (27%)        | 84×     |
| Tree-CRF           | 21           | 14     | (66%)        | 5×      |
| PCFG               | 51           | 36     | (70%)        | 1×      |
| Projective CRF     | 70           | 54     | (77%)        | 3×      |
| Non-Projective CRF | 60           | 8      | (16%)        | 71×     |

Table 1: Comparison against Torch-Struct with respect to lines of code for log-partition and relative speedup in the computation of marginal probabilities.

All SynJax distributions inherit from Equinox modules (Kidger and Garcia, 2021) which makes them simultaneously PyTrees and dataclasses. Thereby all SynJax distributions can be transformed with `jax.vmap` and are compatible with any JAX neural framework, e.g. Haiku and Flax.

## 10 Comparison to alternative libraries

JAX has a couple of libraries for probabilistic modeling. Distrax (Babuschkin et al., 2020) and Tensorflow-Probability JAX substrate (Dillon et al., 2017) provide continuous distributions. NumPyro (Phan et al., 2019) and Oryx provide probabilistic programming. DynaMax (Chang et al., 2022) brings state space models to JAX and includes an implementation of HMMs.

PGMax (Zhou et al., 2023) is a JAX library that supports inference over arbitrary factor graphs by using loopy belief propagation. After the user builds the desired factor graph, PGMax can do automatic inference over it. For many structured distributions building a factor graph is the difficult part of implementation because it may require a custom algorithm (e.g. CKY or Needleman–Wunsch). SynJax implements those custom algorithms for each of the supported structures. With SynJax the user only needs to provide the parameters of the distribution and SynJax will handle *both* building of the factor graph and inference over it. For all the included distributions, SynJax also provides some features not covered by PGMax, such as unbiased sampling, computation of entropy, cross-entropy and KL divergence.

Optax (Babuschkin et al., 2020) provides CTC loss implementation for JAX but without support for computation of optimal alignment, marginals over alignment links, sampling alignments etc.

All the mentioned JAX libraries focus on continuous or categorical distributions and, with the exception of HMMs and CTC loss, do not contain distributions provided by SynJax. SynJax fills

this gap in the JAX ecosystem and enables easier construction of structured probability models.

The most comparable library in terms of features is Torch-Struct (Rush, 2020) that targets PyTorch as its underlying framework. Torch-Struct, just like SynJax, uses automatic differentiation for efficient inference. We will point out here only the main differences that would be of relevance to users. SynJax supports larger number of distributions and inference algorithms and provides a unified interface to all of them. It also provides reproducible sampling through controlled randomness seeds. SynJax has a more general approach to computation of entropy that does not depend on semirings and therefore applies to all distributions. SynJax is fully implemented in Python and compiled with `jax.jit` and `numba.jit` while Torch-Struct does not use any compiler optimizations except a custom CUDA kernel for semiring matrix multiplication. If we compare lines of code and speed (Table 1) we can see that SynJax is much more concise and faster than Torch-Struct (see Appendix A for details).

SynJax also provides the fastest and most feature rich implementation of spanning tree algorithms. So far the most competitive libraries for spanning trees were by Zmigrod et al. and Stanojević and Cohen. SynJax builds on Stanojević and Cohen code and annotates it with Numba instructions which makes it many times faster than any other alternative (see Figure 3 in the appendix).

## 11 Conclusion

One of the main challenges in creating deep neural models over structured distributions is the difficulty of their implementation on modern hardware accelerators. SynJax addresses this problem and makes large scale training of structured models feasible and easy in JAX. We hope that this will encourage research into finding alternatives to auto-regressive modeling of structured data.

## Limitations

SynJax is quite fast, but there are still some areas where the improvements could be made. One of the main speed and memory bottlenecks is usage of big temporary tensors in the dynamic programming algorithms needed for computation of log-partition function. This could be optimized with custom kernels written in Pallas.<sup>2</sup> There are some speed gains

<sup>2</sup><https://jax.readthedocs.io/en/latest/pallas>

that would conceptually be simple but they depend on having a specialized hardware. For instance, matrix multiplication with semirings currently does not use hardware acceleration for matrix multiplication, such as TensorCore on GPU, but instead does calculation with regular CUDA cores. We have tried to address this with log-einsum-exp trick (Peharz et al., 2020) but the resulting computation was less numerically precise than using a regular log-semiring with broadcasting. Maximum spanning tree algorithm would be much faster if it could be vectorized – currently it’s executing as an optimized Numba CPU code.

## Acknowledgements

We are grateful to Chris Dyer, Aida Nematzadeh and other members of language team in Google DeepMind for early comments on the draft of this work. We appreciate Patrick Kidger’s work on Equinox and Jaxtyping that made development of SynJax much easier. We also appreciate that Sasha Rush open-sourced Torch-Struct, a library that influenced many aspects of SynJax.

## References

- Ossama Abdel-Hamid, Li Deng, Dong Yu, and Hui Jiang. 2013. *Deep segmental neural networks for speech recognition*. In *Interspeech*, volume 36.
- Wilker Aziz. 2015. *Grasp: Randomised Semiring Parsing*. *Prague Bulletin of Mathematical Linguistics*, 104:51–62.
- Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. 2020. *The DeepMind JAX Ecosystem*.
- Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. *Interpretable neural predictions with differentiable binary variables*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.
- Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima’an. 2017. *Graph convolutional encoders for syntax-aware neural machine*

- translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.
- Yonatan Bisk and Ke Tran. 2018. [Inducing grammars with and for neural machine translation](#). In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 25–35, Melbourne, Australia. Association for Computational Linguistics.
- Oscar Chang, Dongseong Hwang, and Olivier Siohan. 2023. [Revisiting the Entropy Semiring for Neural Speech Recognition](#). In *The Eleventh International Conference on Learning Representations*.
- Peter Chang, Giles Harper-Donnelly, Aleya Kara, Xinglong Li, Scott Linderman, and Kevin Murphy. 2022. [Dynamax](#).
- Do Kook Choe and Eugene Charniak. 2016. [Parsing as language modeling](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.
- Shay B. Cohen, Giorgio Satta, and Michael Collins. 2013. [Approximate PCFG parsing using tensor decomposition](#). In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 487–496, Atlanta, Georgia. Association for Computational Linguistics.
- Charles J. Colbourn, Wendy J. Myrvold, and Eugene Neufeld. 1996. [Two Algorithms for Unranking Arborescences](#). *Journal of Algorithms*, 20(2):268–281.
- Caio Corro and Ivan Titov. 2019. [Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder](#). In *International Conference on Learning Representations*.
- Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. 2008. [On the computation of the relative entropy of probabilistic automata](#). *International Journal of Foundations of Computer Science*, 19(01):219–242.
- David F Crouse. 2016. [On implementing 2D rectangular assignment algorithms](#). *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696.
- Marco Cuturi and Mathieu Blondel. 2017. [Soft-DTW: A Differentiable Loss Function for Time-Series](#). In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, page 894–903. JMLR.org.
- Adnan Darwiche. 2003. [A Differential Approach to Inference in Bayesian Networks](#). *J. ACM*, 50(3):280–305.
- Nathan Day, Andrew Hemmaplardh, Robert E. Thurman, John A. Stamatoyannopoulos, and William S. Noble. 2007. [Unsupervised segmentation of continuous genomic data](#). *Bioinformatics*, 23(11):1424–1426.
- Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. 2017. [TensorFlow Distributions](#).
- Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. [Recurrent neural network grammars](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.
- Jason Eisner. 1996. [Three new probabilistic models for dependency parsing: An exploration](#). In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.
- Jason Eisner. 2002. [Parameter estimation for probabilistic finite-state transducers](#). In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Jason Eisner. 2016. [Inside-Outside and Forward-Backward Algorithms Are Just Backprop \(tutorial paper\)](#). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.
- Yao Fu, Chuanqi Tan, Bin Bi, Mosha Chen, Yansong Feng, and Alexander M. Rush. 2020. [Latent template induction with gumbel-crfs](#). In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA. Curran Associates Inc.
- Joshua Goodman. 1999. [Semiring Parsing](#). *Computational Linguistics*, 25(4):573–606.
- Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. [Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks](#). In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376.
- Andreas Griewank. 1992. [Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation](#). *Optimization Methods and Software*, 1(1):35–54.
- Awni Hannun. 2017. [Sequence Modeling with CTC](#). *Distill*. <https://distill.pub/2017/ctc>.
- Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. [Cooperative learning of disjoint syntax and semantics](#). 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 1118–1128, Minneapolis, Minnesota. Association for Computational Linguistics.

- Julia Hockenmaier, Aravind K Joshi, and Ken A Dill. 2007. [Routes are trees: the parsing perspective on protein folding](#). *Proteins: Structure, Function, and Bioinformatics*, 66(1):1–15.
- Liang Huang, He Zhang, Dezhong Deng, Kai Zhao, Kaibo Liu, David A Hendrix, and David H Mathews. 2019. [LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search](#). *Bioinformatics*, 35(14):i295–i304.
- Rebecca Hwa. 2000. [Sample selection for statistical grammar induction](#). In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, Hong Kong, China. Association for Computational Linguistics.
- Patrick Kidger and Cristian Garcia. 2021. [Equinox: neural networks in JAX via callable PyTrees and filtered transformations](#).
- Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. [Structured attention networks](#). In *International Conference on Learning Representations*.
- Yoon Kim, Chris Dyer, and Alexander Rush. 2019. [Compound probabilistic context-free grammars for grammar induction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.
- Nikita Kitaev, Steven Cao, and Dan Klein. 2019. [Multilingual constituency parsing with self-attention and pre-training](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.
- Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. [Structured Prediction Models via the Matrix-Tree Theorem](#). In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.
- Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. [Dynamic programming algorithms for transition-based dependency parsers](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. [Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data](#). In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. [Numba: A llvm-based python jit compiler](#). In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15*, New York, NY, USA. Association for Computing Machinery.
- Zhifei Li and Jason Eisner. 2009. [First- and second-order expectation semirings with applications to minimum-risk training on translation forests](#). In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore. Association for Computational Linguistics.
- Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. [Segmental recurrent neural networks for end-to-end speech recognition](#). In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH 2016)*.
- Chunchuan Lyu and Ivan Titov. 2018. [AMR parsing as graph prediction with latent alignment](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.
- Xuezhe Ma and Eduard Hovy. 2016. [End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.
- André Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mário Figueiredo. 2010. [Turbo parsers: Dependency parsing by approximate variational inference](#). In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA. Association for Computational Linguistics.
- Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. [Learning Latent Permutations with Gumbel-Sinkhorn Networks](#). In *International Conference on Learning Representations*.
- Arthur Mensch and Mathieu Blondel. 2018. [Differentiable dynamic programming for structured prediction and attention](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3462–3471. PMLR.
- Tsvetomila Mihaylova, Vlad Niculae, and André F. T. Martins. 2020. [Understanding the Mechanics of SPIGOT: Surrogate Gradients for Latent Structure Learning](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2186–2202, Online. Association for Computational Linguistics.

- Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning Series. The MIT Press.
- Maria Nädejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. [Predicting target language CCG supertags improves neural machine translation](#). In *Proceedings of the Second Conference on Machine Translation*, pages 68–79, Copenhagen, Denmark. Association for Computational Linguistics.
- S. B. Needleman and C. D. Wunsch. 1970. [A general method applicable to the search for similarities in the amino acid sequence of two proteins](#). *Journal of Molecular Biology*, 48:443–453.
- Mathias Niepert, Pasquale Minervini, and Luca Franceschi. 2021. [Implicit mle: backpropagating through discrete exponential family distributions](#). *Advances in Neural Information Processing Systems*, 34:14567–14579.
- Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. 2020. [Gradient estimation with stochastic softmax tricks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 5691–5704. Curran Associates, Inc.
- Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van Den Broeck, Kristian Kersting, and Zoubin Ghahramani. 2020. [Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits](#). In *Proceedings of the 37th International Conference on Machine Learning, ICML'20*. JMLR.org.
- Du Phan, Neeraj Pradhan, and Martin Jankowiak. 2019. [Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro](#). In *Program Transformations for ML Workshop at NeurIPS 2019*.
- Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolínek. 2020. [Differentiation of Blackbox Combinatorial Solvers](#). In *International Conference on Learning Representations*.
- Lawrence R. Rabiner. 1990. [A tutorial on hidden markov models and selected applications in speech recognition](#). In *Readings in Speech Recognition*, pages 267–296. Elsevier.
- Alexander Rush. 2020. [Torch-Struct: Deep Structured Prediction Library](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.
- Sakakibara, Underwood, Mian, and Haussler. 1994. [Stochastic Context-Free Grammars for Modeling RNA](#). In *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, volume 5, pages 284–293. IEEE.
- Sunita Sarawagi and William W Cohen. 2004. [Semi-markov conditional random fields for information extraction](#). In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.
- Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. [Transformer Grammars: Augmenting Transformer Language Models with Syntactic Inductive Biases at Scale](#). *Transactions of the Association for Computational Linguistics*, 10:1423–1439.
- David A. Smith and Noah A. Smith. 2007. [Probabilistic Models of Nonprojective Dependency Trees](#). In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140, Prague, Czech Republic. Association for Computational Linguistics.
- Miloš Stanojević. 2022. [Unbiased and efficient sampling of dependency trees](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1691–1706, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Miloš Stanojević and Shay B. Cohen. 2021. [A Root of a Problem: Optimizing Single-Root Dependency Parsing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10540–10557, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. [A minimal span-based neural constituency parser](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.
- Charles Sutton and Andrew McCallum. 2007. [An Introduction to Conditional Random Fields for Relational Learning](#). *Introduction to statistical relational learning*, page 93.
- Simo Särkkä and Ángel F. García-Fernández. 2021. [Temporal Parallelization of Bayesian Smoothers](#). *IEEE Transactions on Automatic Control*, 66(1):299–306.
- R. E. Tarjan. 1977. [Finding optimum branchings](#). *Networks*, 7(1):25–35.
- Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. [Unsupervised neural hidden Markov models](#). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX. Association for Computational Linguistics.
- W. T. Tutte. 1984. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley, Menlo Park, CA.

- L.G. Valiant. 1979. [The complexity of computing the permanent](#). *Theoretical Computer Science*, 8(2):189–201.
- 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.
- Zhiyong Wang and Jinbo Xu. 2011. [A conditional random fields method for RNA sequence–structure relationship modeling and conformation sampling](#). *Bioinformatics*, 27(13):i102–i110.
- Ronald J. Williams. 1992. [Simple statistical gradient-following algorithms for connectionist reinforcement learning](#). *Mach. Learn.*, 8(3–4):229–256.
- David Bruce Wilson. 1996. [Generating Random Spanning Trees More Quickly than the Cover Time](#). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*, page 296–303, New York, NY, USA. Association for Computing Machinery.
- Songlin Yang, Wei Liu, and Kewei Tu. 2022. [Dynamic programming in rank space: Scaling structured inference with low-rank HMMs and PCFGs](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4797–4809, Seattle, United States. Association for Computational Linguistics.
- Guangyao Zhou, Antoine Dedieu, Nishanth Kumar, Wolfgang Lehrach, Miguel Lázaro-Gredilla, Shrini Kushagra, and Dileep George. 2023. [Pgmax: Factor graphs for discrete probabilistic graphical models and loopy belief propagation in jax](#).
- Ran Zmigrod, Tim Vieira, and Ryan Cotterell. 2020. [Please Mind the Root: Decoding Arborescences for Dependency Parsing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4809–4819, Online. Association for Computational Linguistics.
- Ran Zmigrod, Tim Vieira, and Ryan Cotterell. 2021. [Efficient Computation of Expectations under Spanning Tree Distributions](#). 9:675–690.

| Distribution       | parameters                 |
|--------------------|----------------------------|
| CTC                | b= 64, n=124, l=512        |
| Alignment CRF      | b= 16, n=256, m=256        |
| Semi-Markov CRF    | b= 1, n= 64, nt= 32, k=8   |
| Tree CRF           | b= 128, n=128, nt=128      |
| Linear-Chain CRF   | b= 128, n=256, nt= 32      |
| PCFG               | b= 1, n= 48, nt= 64, pt=96 |
| HMM                | b= 1, n=128, nt= 32        |
| Non-Projective CRF | b=1024, n=128              |
| Projective CRF     | b= 128, n=128              |

Table 2: Sizes of tested distributions.

## A Empirical comparisons

### A.1 Comparison with Torch-Struct

We compared against the most recent Torch-Struct<sup>3</sup> commit from 30 Jan 2022. To make Torch-Struct run faster we have also installed its specialized kernel for semiring matrix multiplication *genbmm*<sup>4</sup> from its most recent commit from 11 Oct 2021. While Torch-Struct supports some of the same distributions as SynJax we did not manage to do speed comparison over all of them. For example, AlignmentCRF of Torch-Struct was crashing due to mismatch of PyTorch, Torch-Struct and genbmm changes about in-place updates. We compile SynJax with `jax.jit` and during benchmarking do not count the time that is taken for compilation because it needs to be done only once. We also tried to compile Torch-Struct using TorchScript by tracing but that did not work out of the box. Comparisons are done on A100 GPU on Colab Pro+. The results are shown in Table 1 in the main text. Table 2 shows sizes of the distributions being tested.

### A.2 Comparison with Zmigrod et al.

Non-Projective spanning trees present a particular challenge because they cannot be vectorized easily due to dynamic structures that are involved in the algorithm. The main algorithms and libraries for parsing this type of trees are from Zmigrod et al. (2020)<sup>5</sup> and Stanojević and Cohen (2021)<sup>6</sup>. The first one is expressed as a recursive algorithm, while the second one operates over arrays of fixed size in iterative way. This makes Stanojević and Cohen algorithm much more amendable to Numba optimization. We took that code and just annotated

<sup>3</sup><https://github.com/harvardnlp/pytorch-struct>

<sup>4</sup><https://github.com/harvardnlp/genbmm>

<sup>5</sup><https://github.com/rycolab/spanningtrees>

<sup>6</sup><https://github.com/stanojevic/Fast-MST-Algorithm>

it with Numba primitives. This made the algorithm significantly faster, especially for big graphs, as can be seen from Figure 3.



Figure 3: Speed comparison of Non-Projective Spanning Tree libraries.

### A.3 Comparison of Maximum Projective Spanning Tree Algorithms

Eisner’s algorithm is virtually the only projective parsing algorithm actively used, if we do not count the transition based parsers. We have found that replacing Eisner’s algorithm with [Kuhlmann et al. \(2011\)](#) tabulation of arc-hybrid algorithm can provide large speed gains both on CPU and GPU. See Figure 4. In this implementation graph size does not make a big difference because it is implemented in a vectorized way so most operations are parallelized.

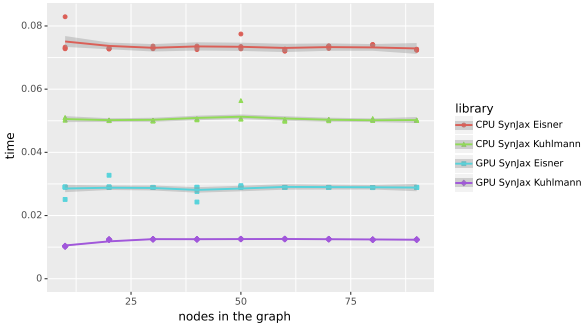


Figure 4: Speed comparison of Projective Maximum Spanning Tree algorithms.



# RESIN-EDITOR: A Schema-guided Hierarchical Event Graph Visualizer and Editor

Khanh Duy Nguyen<sup>1\*</sup>, Zixuan Zhang<sup>1\*</sup>, Reece Suchocki<sup>2</sup>, Sha Li<sup>1</sup>,  
Martha Palmer<sup>2</sup>, Susan Brown<sup>2</sup>, Jiawei Han<sup>1</sup>, Heng Ji<sup>1</sup>

<sup>1</sup>University of Illinois Urbana-Champaign

<sup>2</sup>University of Colorado Boulder

{knguye71, zixuan11, sha12, hanj, hengji}@illinois.edu  
{reece.suchocki, susan.brown, martha.palmer}@colorado.edu

## Abstract

In this paper, we present RESIN-EDITOR, an interactive event graph visualizer and editor designed for analyzing complex events. Our RESIN-EDITOR system allows users to render and freely edit hierarchical event graphs extracted from multimedia and multi-document news clusters with guidance from human-curated event schemas. RESIN-EDITOR's unique features include hierarchical graph visualization, comprehensive source tracing, and interactive user editing, which is more powerful and versatile than existing Information Extraction (IE) visualization tools. In our evaluation of RESIN-EDITOR, we demonstrate ways in which our tool is effective in understanding complex events and enhancing system performance. The source code<sup>1</sup>, a video demonstration<sup>2</sup>, and a live website<sup>3</sup> for RESIN-EDITOR have been made publicly available.

## 1 Introduction

Complex events, such as an international negotiation or a disease outbreak, can take place over a prolonged period of time spanning from weeks to even months. Typically, such complex events can be further deconstructed into many atomic sub-events, where each occurs at a specific time and place. Fully modeling such complex events requires an understanding of the temporal, logical, and hierarchical connections among many sub-events, making the task of event modeling difficult for existing Information Extraction (IE) visualization and analysis tools. Recent research efforts (Wang et al., 2020; Du et al., 2022; Wang et al., 2022) extend the event understanding paradigm to model event-event relations and propose to use event schemas to

guide the organization of complex event structures and predict possible future events. However, convenient visualization and editing tools specifically designed to handle such types of complex events are still largely undeveloped.

A visualization and editing tool is essential for developing and improving an IE system. By utilizing such a tool, system developers can conduct a qualitative error analysis, which involves human analysts carefully investigating any missing or spurious errors, tracing these inaccuracies back to the original documents, and record any proposed modifications or suggestions. Existing tools designed for atomic events (Sheng et al., 2018; Li et al., 2019) are limited when directly applied to complex events, which are much more intricate with complicated event-event relations and multi-document multi-media source inputs. Furthermore, prior analysis tools lack support for interactive user editing, which poses a challenge for model developers to comprehend any real feedback from actual users.

To address the previously discussed challenges and accelerate research of complex events, we present the RESIN-EDITOR, an analysis interface that allows users to visualize, analyze, and freely edit the multi-granularity graphical event narratives extracted from multi-document multimedia news clusters. Compared with existing analysis interfaces, our tool has the following three significant advantages: 1) **Hierarchical Graph Visualization**. Our tool is able to generate a comprehensive visualization of the extracted hierarchical event graph (as shown in Figures 1 and 2). This graph not only includes atomic events and entities, but also includes all event-event hierarchical, temporal, and logical relations of the entire complex event. Such an informative visualization can offer human analysts a rapid and comprehensive understanding of any complex event, significantly improving the efficiency of our error analysis process. Moreover, our graphical visualization is initialized with

\*Equal Contribution.

<sup>1</sup>Data and source code: <https://github.com/blender-nlp/RESIN-Editor>

<sup>2</sup>Video demonstration: <https://www.youtube.com/watch?v=fmW-GwPMrw0>

<sup>3</sup>Live website: <https://blender-nlp.github.io/RESIN-Editor/>

carefully-curated schemas, and the extracted events are grounded to the schema whenever possible, which improves the reliability of our error analysis. 2) **Comprehensive Source Tracing**. Our tool is able to show multimedia provenances for all extracted events, entities, and relations, including source documents, text spans, images, and bounding boxes. These trace-back provenances are presented in a clear-structured and tabular format, which greatly simplifies the users' task of understanding and evaluating the performance of the extraction system. 3) **Interactive User Editing**. Most importantly, with our analysis tool, users are allowed to edit and manipulate almost every extracted element depicted in the visualized event graph and save their edited event graph for downstream usage. This includes not only the node attributes of events and entities, such as names, types, and description sentences but also the graphical structures, like the entity-entity relations and event-event temporal links. Through open and direct interactions with intended users, our tool equips model developers with a comprehensive and precise understanding of actual user requirements.

Our main contributions can be summarized as:

- A novel development to a multimedia analysis tool for complex event understanding, which enables multi-granularity visualization, comprehensive source document tracing, and interactive user editing.
- Extensive empirical studies show that our tool is significantly more effective than existing tools in understanding and analyzing complex events.

## 2 Backend System Overview

The RESIN-EDITOR system provides an interface for event graph visualization with the added capability of graph editing. The visualized graph is an instantiated schema graph, which results from matching an extracted event graph with an event schema. An event schema is a prototypical description of the usual progression of a particular type of complex event, which can be induced from historical news documents.

In this section, we present an overview of how this event graph is generated, with the main components being multimedia event extraction, cross-

document event organization, and event prediction in hierarchical schemas.

**Multimedia Event Extraction** The input to our system is a multimedia news cluster. This cluster includes multiple news documents with related images from a variety of news sources, all describing a single news event. From this cluster, we first perform event extraction for atomic events. Following the convention of ACE-2005<sup>4</sup>, each event is composed of an event trigger as well as several event arguments. Since we would like our system to work for all news scenarios, we employ GLEN (Zhan et al., 2023), the event trigger extraction system that works for 3k+ event types, making it the most comprehensive event detection system to date. We then extract event arguments with a template-based generation approach (Li et al., 2021) distilled from a large language model that is prompted to extract arguments as code structures (Wang et al., 2023). To handle visual input, we use (Li et al., 2020) to extract events from images and merge them with events extracted from text. We represent the atomic events from the multimedia event extraction system as star-shaped graphs such as the *Diagnosis* event shown in Figure 1.

**Cross-Document Event Organization** After extracting atomic events and entities, all event and entity mentions that correspond to the same real-life event and entity are merged through coreference resolution (Lai et al., 2021), and linked to Wikidata through entity linking (Lai et al., 2022). We then conduct event-event temporal relation extraction (Jin et al., 2023) to organize the events into a temporal event graph. We refer to the resulting event graph as the instance event graph, which serves as a semantic representation of the input news cluster in aggregate. The event graph exemplifies unfolding narratives within the given news cluster, as well as connects entities across articles.

**Matching and Prediction with Hierarchical Schemas** After extracting the instance event graph, we perform event instantiation and prediction with the help of event schemas. The schema is typically depicted as a graph in which the nodes are generalized event types linked by inter-event relations, and our task is to instantiate the schema with our instance event graph and make predictions for those unmatched events. As compared

<sup>4</sup><https://www ldc.upenn.edu/collaborations/past-projects/ace>

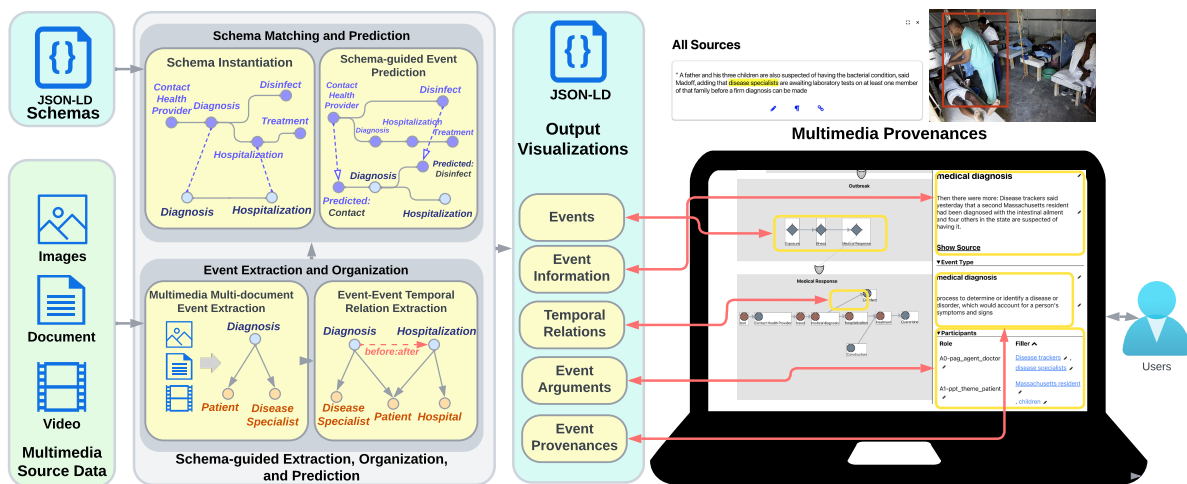


Figure 1: Overview of our RESIN-EDITOR system. Given a multimedia multi-document news cluster and the schema graph, our system extracts the atomic events, organizes them into an instance event graph, performs schema-guided matching and prediction, and finally interactively visualizes the results for user editing.

to previous systems (Du et al., 2022; Wang et al., 2022), we now facilitate event matching and prediction on hierarchical schema structures, where the events in the schema are presented with parent-child relations to represent the different granularity of events. All of the extracted event nodes in our system are grounded in the structured event hierarchy, either by matching to a schema node, or being attached as a child event to a schema node. Event and entity nodes in our extracted event graph are matched top-down, enabling event prediction based on our graphical neural network event prediction system (Wang et al., 2022).

### 3 Graphical Interface

Our graphical interface visualizes the instantiated schema graph after matching and prediction along with the source documents that provided provenance for the extracted events. The interface (Figure 2) is organized in the panel hierarchy illustrated in Figure 3. The graph panel, which is the canvas for showing the event graph, is the primary access point. The overlay panels, such as the entity table, provenance, and mini-map, enhance the exploration experience with contextually relevant information or provide navigation support.

#### 3.1 Interactive Graph Panel

We adopt a graph-based representation for the schema-matching results to provide a streamlined and efficient error analysis interface for event analysts. For each event, we use color coding to illustrate whether the event is extracted from the

source document (source-only), from the schema (predicted), or found in both (matched).

To assist analysts or system developers in making informed decisions, we find it critical to present the relations between events clearly and disentangle different types of relations. We support 3 types of relations among events in our visualization tool:

- **Hierarchical event-event relations.** Parent-child relations between events are visualized by arranging them on the vertical axis: parent events are shown above their children events. Primitive events, which are leaf nodes in the graph, are displayed as circular nodes, whereas parent events are displayed as diamond-shaped nodes. Initially, we only show the top-level events as an overview of the scenario. Users can click on the parent event to expand the graph and dive deeper into the details of the event.
- **Temporal relations.** We order events with temporal relations along the horizontal axis from left to right. In the case of misaligned temporal relations, the user can change the event order by editing the temporal edge.
- **Logical relations.** We support the usage of logical relations defined on a group of events to enforce stronger logical constraints. The *OR gate* signifies that one or more events in the group can occur, the *AND gate* necessitates the occurrence of all events, and the *XOR gate* permits exclusively one event to happen. Given the common use of digital logic gates

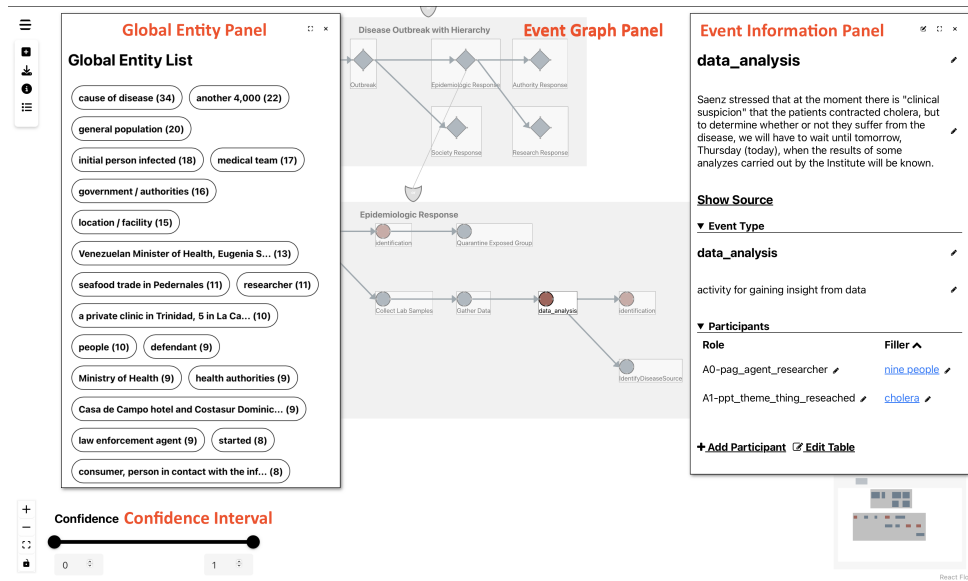


Figure 2: The RESIN-EDITOR interface with *Cholera Decease Outbreak* event graph.

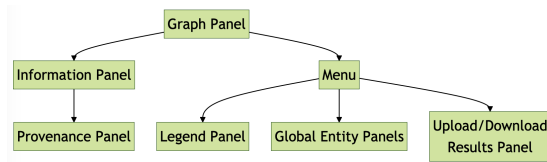


Figure 3: The View Panel Hierarchy illustrates the call sequence of the panels. The most significant panel is located at the top, and subsequent panels are activated from their immediate parent panels.

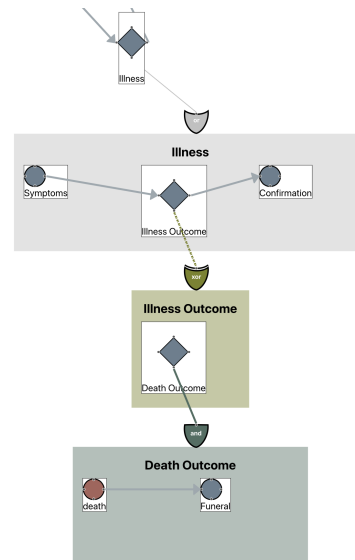


Figure 4: After an *Illness* event, possible subsequent events include *Symptoms*, *Illness Outcomes*, or *Confirmation*. Following the *Illness Outcomes* event, only a single *Death Outcomes* event can occur. The *Death Outcomes* event then invariably leads to both *Death* and *Funeral* events.

in computer science and the conceptual analogy they provide, we employ them to depict logical gates in our event graph (Figure 4).

To improve navigation efficiency in the Graph Panel, we include direct navigation through drag panels and event dragging, a zoom function for detailed exploration, and a minimap for a macroscopic overview of the graph. More detailed information and context can be accessed by clicking on an event node, which triggers an information panel.

### 3.2 Information Panel

The individual observer without assistance has significant limitations regarding how much information they can perceive, process, and retain (Miller, 1956). To avoid interface clutter, we partition the event's information into three distinct sections:

**Event Description** Include the event name and the description of the event. Furthermore, descriptions of events can be viewed swiftly by hovering over the events on the Graph Panel.

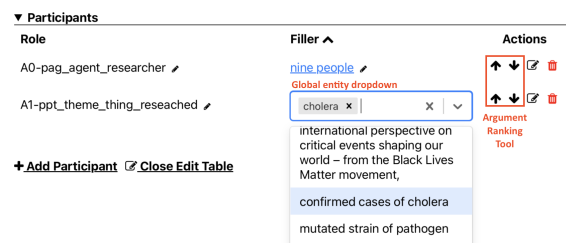


Figure 5: Updating the argument lists of the *Data Analysis* event, incorporating the *Confirmed Cases of Cholera* entity into the theme research.

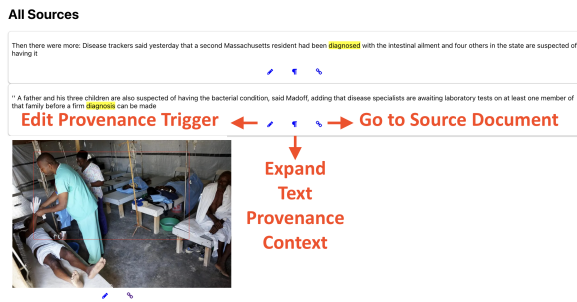


Figure 6: Multimedia provenances support *Disease Specialist* entity.

**Event Type** The event type information is either from the event schema or the event detection system. We use the XPO ontology (Spaulding et al., 2023) for the scope of event types covered by our system. This information might prove helpful in cases where the event description is insufficiently descriptive.

**Argument Table** Each event is associated with multiple argument roles as defined by the ontology, and each argument role can have multiple fillers. We present this information in the argument table which consists of rows of arguments (e.g., location, places, participants) and their associated entities, as shown in Figure 5. We include only the entities backed by sources, leaving out generic ones. In editing mode, the user can reorder the arguments per their needs, ensuring that the most relevant arguments are prioritized at the top. Furthermore, the edit mode provides a comprehensive list of entities from which analysts can add any relevant missing entities as arguments. Each information display includes an editing function, allowing analysts to adjust the presented event information while analyzing schema-matching results.

### 3.3 Provenance Panel

Identifying the source document that validates schema matching is essential for analysts during error analysis because it helps to verify any inaccuracies either from the IE system or in matching process. Hence, in the provenance panel (Figure 6), we display the text and image provenances for source-only events, matched events, and entities. We utilize text highlighting within the paragraph to indicate text provenances, allowing the user to find the relevant citation in the source document easily. Additionally, image provenances employ a bounding box to indicate the referenced entity in the image. Under each piece of provenance, we

provide three functions:

**Edit the provenance mentions.** Analysts can substitute inaccurate text spans in the source document with the correct trigger span for text provenance. For image provenance, the bounding box can be modified to suit the entity in the image better. For instance, in Figure 6, analysts can realign and resize the bounding box to correspond with the nurse image. This adjustment allows for adequately presenting the *Disease Specialist* entity inside the provenance of the *Diagnosis* event.

**Expand the provenance context.** Analysts can expand and view the entire paragraph containing the extracted provenance, but only for text provenance.

**Go to source document.** The feature enables tracing back to the original documents containing the sentence, providing analysts with additional contextual information about the provenance. This tool becomes essential when analysts need to authenticate the provenance’s integrity.

### 3.4 Event Filtering Tools

To ease navigation through numerous graph events, we develop these two event filters:

**Global Entity List** This feature aids analysts in isolating entity-related events by reducing the opacity of unrelated events. For instance, selecting *Cholera* entity in Figure 2 emphasizes the connected *Data Analysis* event, while other events become dim. Moreover, the global entities list ranks entities based on their occurrence in graph events, highlighting those most frequently mentioned across all events.

**Confidence Interval** The schema-matching process between the instance graph and the schema assigns a confidence value to each event from its schema matching and prediction component (Wang et al., 2022). This tool is beneficial when analysts aim to discover matching events with less certainty. By selecting a lower confidence interval, analysts can monitor all events within that range and investigate those with low confidence values.

## 4 Experiments

We present our empirical experiments with multiple document clusters to evaluate the effectiveness of our editor.

**Evaluation Setup** We use two high-impact news scenarios *Disease Outbreak* and *Terrorist Attacks* with four newsworthy complex events, where the detailed statistics are shown in Table 1.

**Results** Based on the results shown in Table 2, RESIN-EDITOR significantly outperforms the traditional method of direct source document analysis, showing considerable improvements across all main IE tasks in our backend system. Such an improvement is particularly notable in the prediction task, probably because the original documents offer limited supporting information for predicting new events. However, our visualizer can provide a well-organized graphical event timeline, which is greatly beneficial for comprehending the events and making accurate predictions.

| News Cluster                | # Docs | # Images |
|-----------------------------|--------|----------|
| Cholera outbreak, Dominica  | 13     | 114      |
| E. coli outbreak, USA       | 16     | 118      |
| Legionnaires outbreak, USA  | 14     | 66       |
| Mogadishu bombings, Somalia | 13     | 57       |

Table 1: Statistics of the news document clusters.

| Tasks                        | Prec | Rec  | F1          |
|------------------------------|------|------|-------------|
| Event Triggers (documents)   | 75.5 | 69.0 | 72.1        |
| Event Triggers (visualizer)  | 84.9 | 76.3 | <b>80.4</b> |
| Event Arguments (documents)  | 68.0 | 58.6 | 63.0        |
| Event Arguments (visualizer) | 84.0 | 72.4 | <b>77.8</b> |
| Schema Matching (documents)  | 67.9 | 62.3 | 65.0        |
| Schema Matching (visualizer) | 71.1 | 64.5 | <b>67.6</b> |
| Prediction (documents)       | 62.9 | 50.0 | 55.7        |
| Prediction (visualizer)      | 93.1 | 66.7 | <b>77.8</b> |

Table 2: The improved IE task performances using our designed visualizer compared with directly looking into the source documents. We focus on the four main IE tasks in our backend system, and the performances are characterized in precision, recall, and F1 scores.

## 5 Related Work

In the IE visualization domain, (Jenkins et al., 2023; Vacareanu et al., 2022) offer event-focused indexed documents, while others (Ma et al., 2021; Ning et al., 2018) use temporal-relation entity graphs. However, both methods lack hierarchical event organization which leads to clutter in the graph display. Additionally, despite presenting entity provenance triggers, they are not able to assist in amending extraction errors.

In terms of schema visualization, Mishra et al. (2021) introduce a graph-based interface tailored

to depict schemas, emphasizing event order and entity relations. While the interface allows users to edit schemas, the user must come up with editing suggestions based on his/her own knowledge without any help from the interface. More recently, Zhang et al. (2023) introduce a human-in-the-loop interface that employs pre-trained LLMs to help create schema graphs detailing event sequences in scenarios. Whereas the use of LLMs can alleviate the manual effort of the user, the LLM-generated schema is not grounded in real news documents and may be prone to hallucination.

Existing tools mainly focus on either IE extraction or schema visualization. Inspired by the RESIN-11 system (Du et al., 2022), which generates schema-guided event graphs from multi-modal news documents, we introduce a unified tool that integrates both the schema and the extracted event graph, providing a comprehensive visualization for complex event analysis. By matching the extracted event graph with the human-curated schema, we are able to effectively organize the complex event and guide users in providing informed feedback on both the IE results and the schema.

## 6 Conclusion and Future Work

To the best of our knowledge, RESIN-EDITOR is the first graphical user interface for editing and visualizing schema-guided event graphs. Our tool assists users by highlighting the matching results, allowing them to fix discrepancies between event attributes and their argument entities and resolve disparities between extracted entities and their multimedia provenance. Our results show that our solution outperforms the traditional method of error straight from their sources in primary IE tasks. While we recognize the importance of comparing with other tools, our current experimentation is limited due to time constraints. We plan to conduct extensive experiments in the future to validate our results further.

In future work, we aim to develop a ranking system that considers the relevance of multimedia provenance so that users can quickly view the most related sentence/image. We also plan to integrate the event grounding tool, so that users can easily add new events by providing natural language descriptions, and the system will provide event type suggestions and fill in the argument template.

## Acknowledgements

This research is based on work supported by U.S. DARPA KAIROS Program No. FA8750-19-2-1004. 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 DARPA, 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

- 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.
- Chris Jenkins, Shantanu Agarwal, Joel Barry, Steven Fincke, and Elizabeth Boschee. 2023. [Massively multi-lingual event understanding: Extraction, visualization, and search](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 247–256, Toronto, Canada. Association for Computational Linguistics.
- Xiaomeng Jin, Haoyang Wen, Xinya Du, and Heng Ji. 2023. [Toward consistent and informative event-event temporal relation extraction](#). In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 23–32, Toronto, ON, Canada. Association for Computational Linguistics.
- Tuan Lai, Heng Ji, Trung Bui, Quan Hung Tran, Franck Dernoncourt, and Walter Chang. 2021. [A context-dependent gated module for incorporating symbolic semantics into event coreference resolution](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3491–3499, Online. Association for Computational Linguistics.
- Tuan Lai, Heng Ji, and ChengXiang Zhai. 2022. [Improving candidate retrieval with entity profile generation for Wikidata entity linking](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3696–3711, 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, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. 2020. [Cross-media structured common space for multimedia event extraction](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2557–2568, Online. Association for Computational Linguistics.
- Sha Li, Heng Ji, and Jiawei Han. 2021. [Document-level event argument extraction by conditional generation](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.
- Mingyu Derek Ma, Jiao Sun, Mu Yang, Kung-Hsiang Huang, Nuan Wen, Shikhar Singh, Rujun Han, and Nanyun Peng. 2021. [EventPlus: A temporal event understanding pipeline](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 56–65, Online. Association for Computational Linguistics.
- George A. Miller. 1956. [The magical number seven, plus or minus two: Some limits on our capacity for processing information](#). *Psychological Review*, 63(2):81–97.
- 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, Online. Association for Computational Linguistics.
- Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018. [CogCompTime: A tool for understanding time in natural language](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–77, Brussels, Belgium. Association for Computational Linguistics.
- Yongpan Sheng, Zenglin Xu, Yafang Wang, Xiangyu Zhang, Jia Jia, Zhonghui You, and Gerard de Melo. 2018. [Visualizing multi-document semantics via open domain information extraction](#). In *Machine*

*Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part III*, volume 11053 of *Lecture Notes in Computer Science*, pages 695–699. Springer.

Elizabeth Spaulding, Kathryn Conger, Anatole Gershman, Rosario Uceda-Sosa, Susan Windisch Brown, James Pustejovsky, Peter Anick, and Martha Palmer. 2023. [The darpa wikidata overlay: Wikidata as an ontology for natural language processing](#). In *Workshop on Interoperable Semantic Annotation (ISA-19)*, page 12.

Robert Vacareanu, George C.G. Barbosa, Enrique Noriega-Atala, Gus Hahn-Powell, Rebecca Sharp, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2022. [A human-machine interface for few-shot rule synthesis for information extraction](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pages 64–70, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. [Joint constrained learning for event-event relation extraction](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 696–706, Online. Association for Computational Linguistics.

Hongwei Wang, Zixuan Zhang, Sha Li, Jiawei Han, Yizhou Sun, Hanghang Tong, Joseph P. Olive, and Heng Ji. 2022. [Schema-guided event graph completion](#). In *The 4th Conference on Automated Knowledge Base Construction (AKBC 2022)*.

Xingyao Wang, Sha Li, and Heng Ji. 2023. [Code4Struct: Code generation for few-shot event structure prediction](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada. Association for Computational Linguistics.

Qiusi Zhan, Sha Li, Kathryn Conger, Martha Palmer, Heng Ji, and Jiawei Han. 2023. [Glen: General-purpose event detection for thousands of types](#).

Tianyi Zhang, Isaac Tham, Zhaoyi Hou, Jiaxuan 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. 2023. [Human-in-the-loop schema induction](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 1–10, Toronto, Canada. Association for Computational Linguistics.



# DRGCODER: Explainable Clinical Coding for the Early Prediction of Diagnostic-Related Groups

Daniel Hajjaligol<sup>1</sup>, Derek Kaknes<sup>2</sup>, Tanner Barbour<sup>2</sup>, Daphne Yao<sup>1</sup>,  
Chris North<sup>1</sup>, Jimeng Sun<sup>3</sup>, David Liem<sup>4</sup>, Xuan Wang<sup>1</sup>

<sup>1</sup>Department of Computer Science, <sup>2</sup>Department of Statistics, Virginia Tech

<sup>3</sup>Department of Computer Science, University of Illinois Urbana-Champaign

<sup>4</sup>Department of Medicine, University of California, Davis

<sup>1,2</sup>{danielhajjaligol, derekkaknes, tannerb3, danfeng, north, xuanw}@vt.edu,

<sup>3</sup>{jimeng}@illinois.edu, <sup>4</sup>{daliem}@ucdavis.edu

## Abstract

Medical claim coding is the process of transforming medical records, usually presented as free texts written by clinicians, or discharge summaries, into structured codes in a classification system such as ICD-10 (International Classification of Diseases, Tenth Revision) or DRG (Diagnosis-Related Group) codes. This process is essential for medical billing and transitional care; however, manual coding is time-consuming, error-prone, and expensive. To solve these issues, we propose DRGCODER<sup>1,2</sup>, an explainability-enhanced clinical claim coding system for the early prediction of medical severity DRGs (MS-DRGs), a classification system that categorizes patients' hospital stays into various DRG groups based on the severity of illness and mortality risk. The DRGCODER framework introduces a novel multi-task Transformer model for MS-DRG prediction, modeling both the DRG labels of the discharge summaries and the important, or *salient* words within the discharge summaries. We allow users to inspect DRGCODER's reasoning by visualizing the weights for each word of the input. Additionally, DRGCODER allows users to identify diseases within discharge summaries and compare across multiple discharge summaries.

## 1 Introduction

Inpatient care, defined as care for hospital patients who stay overnight, are one of the biggest components in healthcare costs, accounting for approximately 31% of total medical expenses (Muka et al., 2015). Appropriate determination of costs associated with inpatient care are based on assigning one (of potentially many) ICD or DRG codes to a given patient. This is a crucial process for medical insurance billing and healthcare improvement, but it can be very time-consuming, prone to errors,

<sup>1</sup>Our demo is available at <https://huggingface.co/spaces/danielhajjaligol/DRGCoder>

<sup>2</sup>A video demonstrating the demo can be found at <https://www.youtube.com/watch?v=pcdiG6Vwq1A>

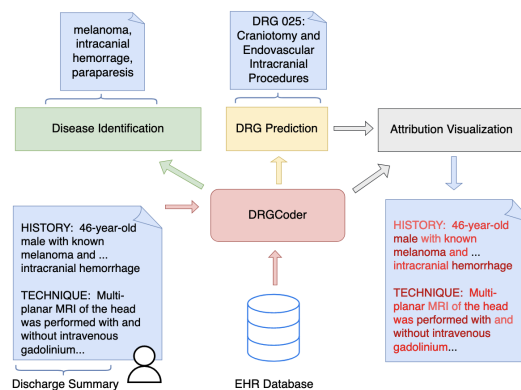


Figure 1: Overview of DRGCODER. Given a discharge summary, DRGCODER (1) identifies diseases, (2) predicts the corresponding DRG, and (3) highlights the importance of each word in the discharge summary contributing to the DRG prediction.

and expensive when done manually. On average, it takes a medical coder about 20 minutes to code a single inpatient stay, and with 35 million inpatient stays in the United States each year, manual coding can be a very laborious and expensive process.

The current process for coding inpatient records most often consists of a Certified Inpatient Coder (CIC) manually reviewing a medical record and identifying every codable entity within that text. Often times, this process is completed across two completely separate software applications: the medical record is either accessed via the electronic health record (EHR) system or else “printed”, usually in a digital (pdf) format, for the coder to review, while the ICD-10 codes are entered into a separate computer-assisted coding (CAC) application. These collection of ICD codes are then used to determine the correct DRG code. Alternatively, early prediction of DRG codes attempt to identify DRG codes directly from the discharge summary, bypassing ICD code prediction.

There have been many attempts to automate clinical coding while incorporating explainability. AI-

though there are many methods that compute attention for each word from the input discharge summary (Dong et al., 2021; Mullenbach et al., 2018; Khalid et al., 2022; Liu et al., 2021), most of them don't utilize the powerful contextual embeddings backed by Transformer (Vaswani et al., 2017) models. Only recently however has there been Transformer-based methodologies for automating clinical coding (Wang et al., 2022; Trigueros et al., 2022). An example of this is Medical Concept Driven Attention (Wang et al., 2022), where they align both clinical notes and Wikipedia documents into topic space via topic modeling. While their architecture can handle any encoder, they claim that using a Transformer as their encoder produces inferior results, as documents are too long. Our work is most similar to Trigueros et al., where they use a multi-task Transformer-based approach, except they perform entity-linking on medical concepts related to ICD codes. Despite advancements in both Wang et al.; Trigueros et al., they don't take into account salient words: words that are *explicitly* important for clinical coding.

To address these issues, we propose DRGCODER (Figure 1), an explainable clinical coding system that provides both a predicted MS-DRG code and highlighted areas of interest within the discharge summary text to support medical coders. DRGCODER first predicts the appropriate DRG based on a novel multi-task Transformer architecture that incorporates both discharge summaries and salient words within the summaries. While there are many types of DRG codes, we choose to focus on identifying MS-DRG codes, as the MS-DRG system is the most widely used, standardized DRG system, and it is maintained by a single entity (The Centers for Medicare & Medicaid Services). DRGCODER allows users to gain better insight into MS-DRG classification by highlighting the areas of the discharge summary that explain the prediction. Furthermore, DRGCODER identifies diseases within discharge summaries and functionality to compare DRGCODER results across multiple discharge summaries.

1. We propose DRGCODER, an explainable clinical claim coding system for the early prediction of MS-DRGs. DRGCODER is supported by a novel multi-task text classification that learns to identify MS-DRG codes and important word.

2. We visualize the importance each word has on the MS-DRG prediction via a heatmap, allowing for researchers to understand DRGCODER predictions.
3. Information extraction processes, such as named-entity recognition and dense passage retrieval are incorporated to identify diseases within a discharge summary and find related discharge summaries, respectively.

## 2 Related Work

### 2.1 Early DRG Prediction

The early DRG prediction literature is shallow. Numerous machine learning techniques, such as Naive Bayes, Bayesian Networks, and Decision Trees have been applied for early DRG prediction (Gartner et al., 2015); however, they hand-crafted many of their features. In contrast, DRGCODER automatically learns features through the usage of contextualized embeddings. The framework proposed in Liu et al. applies the Convolutional Attention for Multi-Label classification (CAML) framework (Mullenbach et al., 2018) to MS-DRG codes and additionally retrieves the most important input words based on the prediction. Their framework, unlike DRGCODER, does not leverage the power of contextualized embeddings via Transformer models.

### 2.2 Canonical DRG Prediction

Canonical DRG prediction systems require the availability of ICD codes for each patient. Unfortunately, many of these systems are closed-source, as many are commercialized<sup>3,4,5</sup>; however, there are open-source models that have been designed to tackle this problem. DEEPDRG (Islam et al., 2021) utilizes gated recurrent units (GRUs) to predict DRG codes; however, the authors only focus on DRG codes related to urinary diseases. This significantly impacted the granularity of the classification problem, resulting in the label space consisting of only 200 DRG codes. AMANet (He et al., 2020) predicts DRGs by viewing the input data from a multi-view perspective; each input is represented by the given diagnosis and procedures of a given patient. KG-MTT-BERT (He et al., 2022) is a multi-modal model that embeds clinical notes using BERT (Devlin et al., 2018) and a custom

<sup>3</sup><https://nym.health/>

<sup>4</sup><https://www.3m.com/>

<sup>5</sup><https://www.artificialmed.com/>

Table 1: Statistics for the MS-DRG-related data from MIMIC-III. Note that DRGCODER only utilizes MS-DRG patient-related data.

|                       |        |
|-----------------------|--------|
| <b>Patients</b>       | 18,132 |
| <b>Hospital Stays</b> | 21,440 |
| <b>Unique DRGs</b>    | 738    |

medical knowledge graph. Both KG-MTT-BERT and AMANet, however, do not provide explainable architectures, unlike DRGCODER.

### 2.3 Explainable Clinical Coding

Previous research has utilized label-wise attention mechanisms to highlight key elements such as n-grams (Mullenbach et al., 2018), words, and sentences (Dong et al., 2021) during the coding process. However, there is a need for further research to evaluate their effectiveness and incorporate more inherently explainable methods. Additionally, current methods primarily focus on explaining model decisions by analyzing model attention, and there have been observations of correlations between model attention and human attention (Atanasova et al., 2020; Sen et al., 2020; López-García et al., 2023; Wang et al., 2022; Trigueros et al., 2022; Khalid et al., 2022; Liu et al., 2022). Nevertheless, there have been limited research efforts to actively align model attention with human attention during the training of machine learning models.

## 3 System Description

DRGCODER can be broken down into two categories: automated clinical coding and explanation visualization. The former component is home to our MS-DRG classification framework while the latter contains our related summary extraction and disease named-entity recognition modules.

### 3.1 Automated Clinical Coding

Given a discharge summary, our automated clinical coding module identifies salient words located in discharge summaries and predicts MS-DRG codes. For MS-DRG code classification, we develop a novel multi-task BERT-based Transformer model that incorporates both discharge summaries and salient words.

**Database** The external database used for DRG classification is MIMIC (Johnson et al., 2018), which consists of health-related data for over

40,000 patients of the Beth Israel Deaconess Medical Center between 2001 and 2012. In addition to EHRs, MIMIC also contains billing system information, mainly comprising of ICD and DRG codes. The clinical information for each patient is captured in a discharge summary, which has an associated DRG code and (at least one) ICD code. We utilize the discharge summary and MS-DRG code across all components in our system. Although there are over 40,000 patients, only around 18,000 patients have MS-DRG data. Table 1 displays MS-DRG data statistics from MIMIC.

**DRG Classification** DRGCODER predicts a DRG code given a discharge summary and salient words from the discharge summary, which should play a strong role in determining the corresponding DRG code. Ideally, we would have a gold standard dataset as part of our training process: discharge summaries with salient words identified by clinical experts; however, as manual clinical coding is labor intensive, no such dataset exists, to the best of our knowledge. Thus, we chose to operate under the weakly-supervised paradigm by automatically extracting words indicative of ICD codes using BioPortal (Noy et al., 2009), a database of biomedical ontologies. Note that DRGCODER still falls under the early DRG prediction paradigm as we don't use the ICD codes identified by the CICs.

We then input both discharge summaries and ICD concepts into a novel multi-task BERT-based Transformer that learns to jointly identify DRG codes and salient words. Figure 2 demonstrates an example of this module at play. BioPortal identifies the ICD concepts "chest pain", "pain", "edema", "cough", "abdominal pain", and "diarrhea", which are then used as part of the DRG classification process. Additionally, we include external links to both identified ICD concepts and DRG codes for users to explore further.

**Algorithm Design** We propose an explanation-enhanced clinical coding method that aligns the model attention with human attention during training (Fig. 3). In addition to predicting the DRG code of a discharge summary, we introduce an auxiliary task that attempts to predict the salient tokens in the discharge summary. More formally, given a discharge summary  $S = \{t_i | \forall i \in [1, |S|]\}$  composed of tokens  $t_i$  and corresponding saliency labels  $A = \{a_i | \forall i \in [1, |S|]\}$ , where  $a_i = 1$  if token  $t_i$  is salient and 0 otherwise, we jointly predict

## DRGCoder

This interface outlines DRGCoder, an explainable clinical coding for the early prediction of diagnostic-related groups (DRGs). Please note all summaries will be truncated to 512 words if longer.

Input Discharge Summary Here

Assessment: Action: Response: Plan: HPI: 86 yof COPD, CAD, AFib, Crohns Disease, h/o Colon CA, who reportedly was in her USOH until 2-3 days PTA when experienced left pleuritic chest pain and dark and bloody stools. Left-sided chest pain described as sharp, intermittent, and radiating to the back of her neck, improved with supine position, and worse with inspiration. Also notes increasing exertional dyspnea, bilateral leg edema and non-productive cough. In addition, noted dark colored stools mixed with fresh blood, without abdominal pain, N/V, diarrhea, jaundice. ER evaluation revealed BP=130/65, P=108, RR=28, SaO2=87% RA improved to 97% on 4L. Asymmetric swelling of legs. Stool reported as dark and guaiac positive, with streaks of fresh blood. ECG demonstrated A-Fib without acute changes. NGT lavage reportedly negative for blood. CT Angio without evidence for pulmonary embolism, but demonstrated diffuse scattered ground glass. ABD CT demonstrated diverticulosis. Received NTG and DilT for possible CHF exacerbation, and Levaquin for possible CAP. Admitted to the MICU for further iatopm and management.

Examples

HEAD CT: Head CT showed no intracranial hemorrhage or mass effect, but old infarction consistent with past medical history.

Radiologic studies also included a chest CT, which confirmed cavitory lesions in the left lung apex consistent with infectious tuberculosis. This also moderate-sized left pleural effusion.

We have discharged Mrs Smith on regular oral Furosemide (40mg OD) and we have requested an outpatient ultrasound of her renal tract which will be performed in the next few weeks. We will review Mrs Smith in the Cardiology Outpatient Clinic in 6 weeks time.

Blood tests revealed a raised BNP. An ECG showed evidence of left-ventricular hypertrophy and echocardiography revealed grossly impaired ventricular function (ejection fraction 35%). A chest X-ray demonstrated bilateral pleural effusions, with evidence of upper lobe diversion.

Mrs Smith presented to A&E with worsening shortness of breath and ankle swelling. On arrival, she was tachypnoeic and hypoxic (oxygen saturation 82% on air). Clinical examination revealed reduced breath sounds and dullness to percussion in both lung bases. There was also a significant degree of lower limb oedema extending up to the mid-thigh bilaterally.

Submit

| Predicted DRG         | Word Importance  | Diseases  | ICD Concepts   |
|-----------------------|--|---|--|
| G.I. HEMORRHAGE W MCC | assessment, action, response, plan, hpi, 86 yof copd, cad, afib, crohns disease, h/o colon ca, who reportedly was in her usoh until 2-3 days pta, when experienced left pleuritic chest pain and dark and bloody stools, left sided chest pain described as sharp, intermittent, and radiating to the back of her neck, improved with supine position and worse with inspiration, also notes increasing exertional dyspnea, bilateral leg edema and non-productive cough, in addition, noted dark colored stools mixed with fresh blood, without abdominal pain, n/v, diarrhea, jaundice, er evaluation revealed bp=130/65, p=108, rr=28, saoz=87% ra improved to 97% on 4l, asymmetric swelling of legs, stool reported as dark and guaiac positive, with streaks of fresh blood, ecg demonstrated a-fib without acute changes, ngt lavage reportedly negative for blood, ct angio without evidence for pulmonary embolism, but demonstrated diffuse scattered ground glass, abd ct demonstrated diverticulosis, received ngt and dilT for possible chf exacerbation, and levaquin for possible cap, admitted to the micu for further iatopm and management | jaundice, cap, chest pain, chf, abdominal pain, diverticulosis, copd, diarrhea, leg edema, cad, swelling, pulmonary embolism, dyspnea, crohns disease | CHEST PAIN, PAIN, EDEMA, COUGH, ABDOMINAL PAIN, DIARRHEA |

Input Correct DRG:

Input Salient Words (comma separated):

Remove DRG Results

Figure 2: An example of DRG CODER using the given discharge summary. We identify the predicted DRG and ICD concepts, diseases, and attributions (the darker the highlighted token is, the more important it is, and vice-versa.). Additionally, we allow the user to input the correct DRG and important tokens, if available.

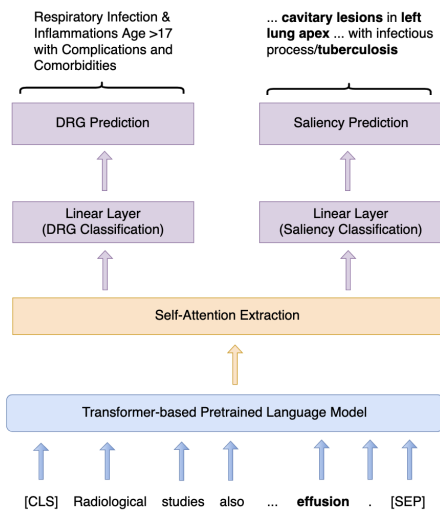


Figure 3: DRG CODER's DRG Prediction algorithm. A discharge summary and salient tokens ("effusion", in this example) are given as input. A Transformer-based language model embeds the discharge summary. For DRG prediction, these embeddings are mapped to a linear layer to get the final prediction. For saliency prediction, the self-attention matrix from the last layer and the last attention head is extracted from the Transformer model. This matrix is then mapped by a linear layer to get (binary) saliency predictions for each token.

the DRG code of the sequence as well as salient tokens.

We use a Clinical Transformer-based pre-trained language model as our backbone (Alsentzer et al., 2019), which has been pretrained on clinical data. For DRG prediction, we employ a cross-entropy loss function  $l_{DRG}$ , where the model prediction is generated by applying a linear classification layer on top of the contextualized embeddings. Saliency classification, on the other hand, utilizes a binary cross-entropy loss function  $l_S$ , with the model prediction obtained by applying a linear classification layer on the last Transformer layer and last attention head for each token. To combine both tasks, we define the final multi-task loss function as a linear combination of the aforementioned two loss functions, resulting in  $l = l_{DRG} + \lambda l_S$ , where  $\lambda$  controls the trade-off between the two tasks.

### 3.2 Explanation Visualization

Users are able to extract further insight into discharge summaries through our attribution visualization, related summary extraction modules. We offer functionality to visualize word importance for the most confident DRG prediction, find related discharge summaries, and extract diseases living in

a queried discharge summary.

**Attribution Visualization** Given that misclassifying DRG codes could lead to revenue loss (Ayub et al., 2019; Zafirah et al., 2018), researchers may want to understand why a MS-DRG code was predicted for a given discharge summary. DRGCODER employs a heat-map on the weights of the DRG Classification module to visualize the most important words in the input. The darker the highlighted word is, the more important the word is. In Figure 2, for example, DRGCODER identified salient words for the input discharge summary was "assessment", "angio", "lavage", "guaiaic positive", etc. DRGCODER is able to understand that the discharge summary is correctly classified as "gastrointestinal hemorrhage". This is because "angio", short for "angiogram", and lavage are both acceptable techniques in identifying gastrointestinal bleeding (Kim et al., 2014; Ousterhout and Feller, 1968). This allows for users to better understand the reasoning for the DRG prediction DRGCODER makes.

**Related Summary Extraction** DRGCODER supports comparison of discharge summary results (predicted DRG, word attribution, diseases, ICD concepts) across *similar* discharge summaries. The degree of similarity between a pair of discharge summaries is dependent upon BioSimCSE (raj Kanakarajan et al., 2022), which trained biomedical sentence embeddings for sentence similarity using SimCSE (Gao et al., 2021). Sentence embeddings were computed by employing contrastive learning (Eq. 1), a training framework that attempts to learn an embedding space where embeddings of similar entities,  $z_i$  and  $z_j$ , are learned to be close together, for some similarity metric (typically cosine similarity). SimCSE and BioSimCSE chose the pairing of  $(z_i, z_j)$  to correspond to the same entity,  $x_i$ , but they input  $x_i$  to a Transformer *twice* in order for the embeddings to obtain different dropout masks (as the dropout probability is random for each embedding).

$$l_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j))}{\sum_{k=1, k \neq i}^N \exp(\text{sim}(z_i, z_k))} \quad (1)$$

Figures 4a and 4b illustrate the related summary extraction module. In Figure 4a, users are displayed the five most similar discharge summaries to the query discharge summary (same as the example in Figure 2), with the similarity scores listed in

Table 2: Macro and Micro AUC and F1 scores of Clinical-BERT and CAML baselines against DRGCODER on the MIMIC-III dataset. For our experiments, we set  $\lambda = 0.5$ .

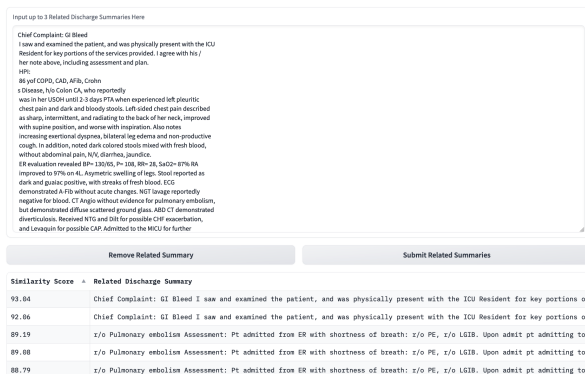
| Model         | Macro-AUC    | Micro-AUC    | Macro-F1     | Micro-F1     |
|---------------|--------------|--------------|--------------|--------------|
| CAML          | 0.871        | 0.956        | 0.084        | 0.270        |
| Clinical-BERT | 0.901        | 0.962        | <b>0.106</b> | 0.256        |
| DRGCODER      | <b>0.911</b> | <b>0.969</b> | 0.101        | <b>0.279</b> |

the beginning of each discharge summary. The user can select multiple related discharge summaries to compare with the original query discharge summary, exemplified in Figure 4b. Users can compare and contrast the DRG prediction, word attribution, diseases, and ICD concepts identified across different discharge summaries. Discharge summaries that have similar diseases and/or salient words, for example, are more likely to be categorized under the same DRG code, or at the very least be classified to related DRG codes. This is also true when comparing ICD concepts, as ICD concepts are more granular than DRG codes, implying that if two summaries have similar ICD concepts, their corresponding DRG codes could be the same/related.

**Disease Named-Entity Recognition** Since the MS-DRG coding system takes into account medical severity of a patient’s condition, it is intuitive that diseases play a pivotal role in DRG prediction. Thus, DRGCODER provides functionality for identifying the diseases associated with a given discharge summary. This module depends on bioNLP (Alonso Casero, 2021), a system that identifies diseases from biomedical text. Figure 2 shows an example of interface for displaying these results. Given the query in Figure 2, DRGCODER lists the diseases between the "Word Importance" and "ICD Concepts" columns, namely "jaundice", "diverticulosis", "diarrhea", etc. Bleeding in the upper gastrointestinal tract has been shown to occur when patients have jaundice (Dixon et al., 1984), while diverticulosis and diarrhea are both conditions that have also been known to result in bleeding throughout the gastrointestinal tract. While this module is not part of the DRG classification process, we include this functionality to allow users to parse the discharge summary quicker so that users can validate DRG predictions faster.

## 4 Evaluation

We illustrate the effectiveness of DRGCODER by comparing the performance of Clinical-BERT and



(a) Related discharge summary pool.



(b) Discharge summary results comparison

Figure 4: (a) Related discharge summary pool. Users can click multiple related discharge summaries to input in the box on the top left. The similarity scores for each related discharge summary are displayed at the beginning of each related summary. (b) Results comparison across multiple discharge summaries.

CAML (Liu et al., 2021) on the MS-DRG portion of the MIMIC-III dataset. We evaluate our performance using F1 and AUC (area under curve) performance metrics. Table 2 indicates that the DRGCODER system outperforms all other compared frameworks. Specifically, when compared to Clinical-BERT, it demonstrates the effectiveness of using salient words as input, making the framework simultaneously explainable and more performant.

## 5 Conclusion

We offer DRGCODER, an explainability-enhanced inpatient claim coding system for MS-DRG code prediction. DRGCODER allows users to visualize words deemed important by our framework, as well as identify diseases and ICD concepts. Furthermore, we offer functionality for similar summary retrieval and comparison across summaries.

## 6 Future Work

We believe a natural extension of DRGCODER would be to incorporate human feedback. We ideally would like to gather feedback about correct DRGs and important words from users who use DRGCODER. Additionally, we would like to effectively incorporate the hierarchy of DRG codes. Although there aren't many levels in the DRG taxonomy, taking into account parent and sibling nodes (DRG codes) and finding appropriate information via knowledge graphs could offer better insight on the subtleties between adjacent DRG codes. For

billing purposes, it would still be insightful for an incorrect DRG prediction to be classified under the same group of DRG codes, as this would indicate similar billing between the incorrectly predicted and ground truth DRG codes.

## 7 Limitations

A limitation of this work is the sole reliance on The MIMIC-III database. While this is a popular database, it only contains medical information from one hospital. Using only this database may not capture health-related information that occurs outside of the sampled population.

## 8 Ethics Statement

The usage of MIMIC requires researchers to ethical principles and guidelines that come with using electronic health records. Although MIMIC is de-identified and open-source, it does contain sensitive and confidential patient information collected, and its use requires the utmost consideration for ethical practices. We believe our purpose falls in line with these values, as we only provide functionality for DRG prediction and analysis. The authors have been granted access to MIMIC by the MIT Laboratory for Computational Physiology.

## 9 Acknowledgement

Our work is sponsored by the Commonwealth Cyber Initiative and a generous gift from the Amazon + VT Center for Efficient and Robust ML.

## References

- Álvaro Alonso Casero. 2021. *Named entity recognition and normalization in biomedical literature: a practical case in SARS-CoV-2 literature*. Ph.D. thesis, ETSI\_Informatica.
- Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.
- Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. *arXiv preprint arXiv:2009.13295*.
- Suniah Ayub, Salvatore T Scali, Julie Richter, Thomas S Huber, Adam W Beck, Javairiah Fatima, Scott A Berceci, Gilbert R Upchurch, Dean Arnaoutakis, Martin R Back, et al. 2019. Financial implications of coding inaccuracies in patients undergoing elective endovascular abdominal aortic aneurysm repair. *Journal of Vascular Surgery*, 69(1):210–218.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- JM Dixon, CP Armstrong, SW Duffy, RA Elton, and GC Davies. 1984. Upper gastrointestinal bleeding: a significant complication after surgery for relief of obstructive jaundice. *Annals of surgery*, 199(3):271.
- Hang Dong, Víctor Suárez-Paniagua, William Whiteley, and Honghan Wu. 2021. Explainable automated coding of clinical notes using hierarchical label-wise attention networks and label embedding initialisation. *Journal of biomedical informatics*, 116:103728.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.
- Daniel Gartner, Rainer Kolisch, Daniel B Neill, and Rema Padman. 2015. Machine learning approaches for early drg classification and resource allocation. *INFORMS Journal on Computing*, 27(4):718–734.
- Yong He, Cheng Wang, Nan Li, and Zhenyu Zeng. 2020. Attention and memory-augmented networks for dual-view sequential learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 125–134.
- Yong He, Cheng Wang, Shun Zhang, Nan Li, Zhaorong Li, and Zhenyu Zeng. 2022. Kg-mtt-bert: Knowledge graph enhanced bert for multi-type medical text classification. *arXiv preprint arXiv:2210.03970*.
- Md Mohaimenul Islam, Guo-Hung Li, Tahmina Nasrin Poly, and Yu-Chuan Li. 2021. Deepdrg: Performance of artificial intelligence model for real-time prediction of diagnosis-related groups. In *Healthcare*, volume 9, page 1632. MDPI.
- Alistair EW Johnson, David J Stone, Leo A Celi, and Tom J Pollard. 2018. The mimic code repository: enabling reproducibility in critical care research. *Journal of the American Medical Informatics Association*, 25(1):32–39.
- Mutahira Khalid, Hasan Ali Khattak, Arsalan Ahmad, and Syed Ahmad Chan Bukhari. 2022. Explainable prediction of medical codes through automated knowledge graph curation framework. In *2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 331–336. IEEE.
- Bong Sik Matthew Kim, Bob T Li, Alexander Engel, Jaswinder S Samra, Stephen Clarke, Ian D Norton, and Angela E Li. 2014. Diagnosis of gastrointestinal bleeding: A practical guide for clinicians. *World journal of gastrointestinal pathophysiology*, 5(4):467.
- Jinghui Liu, Daniel Capurro, Anthony Nguyen, and Karin Verspoor. 2021. Early prediction of diagnostic-related groups and estimation of hospital cost by processing clinical notes. *NPJ digital medicine*, 4(1):103.
- Leibo Liu, Oscar Perez-Concha, Anthony Nguyen, Vicki Bennett, and Louisa Jorm. 2022. Hierarchical label-wise attention transformer model for explainable icd coding. *Journal of biomedical informatics*, 133:104161.
- Guillermo López-García, José M Jerez, Nuria Ribelles, Emilio Alba, and Francisco J Veredas. 2023. Explainable clinical coding with in-domain adapted transformers. *Journal of Biomedical Informatics*, 139:104323.
- Taulant Muka, David Imo, Loes Jaspers, Veronica Colpani, Layal Chaker, Sven J van der Lee, Shanthi Mendis, Rajiv Chowdhury, Wichor M Bramer, Abby Falla, et al. 2015. The global impact of non-communicable diseases on healthcare spending and national income: a systematic review. *European journal of epidemiology*, 30:251–277.
- James Mullenbach, Sarah Wiegrefe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. [Explainable prediction of medical codes from clinical text](#). 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 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.
- Natalya F. Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L. Rubin, Margaret-Anne Storey, Christopher G. Chute, and Mark A. Musen.

2009. [BioPortal: ontologies and integrated data resources at the click of a mouse](#). *Nucleic Acids Research*, 37(suppl\_2):W170–W173.
- DK Ousterhout and Irving Feller. 1968. Occult gastrointestinal hemorrhage in burned patients. *Archives of Surgery*, 96(3):420–422.
- Kamal raj Kanakarajan, Bhuvana Kundumani, Abhijith Abraham, and Malaikannan Sankarasubbu. 2022. Biosimcse: Biomedical sentence embeddings using contrastive learning. In *Proceedings of the 13th International Workshop on Health Text Mining and Information Analysis (LOUHI)*, pages 81–86.
- Cansu Sen, Thomas Hartvigsen, Biao Yin, Xiangnan Kong, and Elke Rundensteiner. 2020. Human attention maps for text classification: Do humans and neural networks focus on the same words? In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4596–4608.
- Owen Trigueros, Alberto Blanco, Nuria Lebeña, Arantza Casillas, and Alicia Pérez. 2022. Explainable icd multi-label classification of ehers in spanish with convolutional attention. *International Journal of Medical Informatics*, 157:104615.
- 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.
- Tao Wang, Linhai Zhang, Chenchen Ye, Junxi Liu, and Deyu Zhou. 2022. A novel framework based on medical concept driven attention for explainable medical code prediction via external knowledge. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1407–1416.
- SA Zafirah, Amrizal Muhammad Nur, Sharifa Ezat Wan Puteh, and Syed Mohamed Aljunid. 2018. Potential loss of revenue due to errors in clinical coding during the implementation of the malaysia diagnosis related group (my-drg®) casemix system in a teaching hospital in malaysia. *BMC health services research*, 18(1):1–11.



# CAMRA: Copilot for AMR Annotation

Jon Z. Cai and Shafiuddin Rehan Ahmed and Julia Bonn

Kristin Wright-Bettner and Martha Palmer and James H. Martin

University of Colorado Boulder

jon.z.cai@colorado.edu

## Abstract

In this paper, we introduce CAMRA (Copilot for AMR Annotations), a cutting-edge web-based tool designed for constructing Abstract Meaning Representation (AMR) from natural language text. CAMRA offers a novel approach to deep lexical semantics annotation such as AMR, treating AMR annotation akin to coding in programming languages. Leveraging the familiarity of programming paradigms, CAMRA encompasses all essential features of existing AMR editors, including example lookup, while going a step further by integrating Propbank roleset lookup as an autocomplete feature within the tool. Notably, CAMRA incorporates AMR parser models as coding copilots, greatly enhancing the efficiency and accuracy of AMR annotators. To demonstrate the tool's capabilities, we provide a live demo accessible at: <https://camra.colorado.edu><sup>1</sup>

## 1 Introduction

Abstract Meaning Representation (AMR) stands as one of the most widely embraced formalisms for deep lexical semantic representation within the NLP community. It effectively captures the lexical semantics present in multiple sentences by employing a directed, acyclic graph, wherein graph nodes form predicate-argument structures locally in Neo-Davidsonian fashion (Banarescu et al., 2013). AMR can address both superficial semantic inquiries, encompassing aspects like "Who did what to whom, when, where, and how," as well as the intricate relationships between various events and states. Beyond these merits, AMR offers an invaluable advantage through its transparent symbolic representation of the semantics inherent in natural language text, significantly benefiting tasks reliant on semantic inference and necessitating interpretability.

<sup>1</sup>publish upon acceptance, demo video link: <https://youtu.be/mS3tzDvVaU8>

Over the past decade, NLP researchers have meticulously transcribed tens of thousands of natural sentences into AMR graphs (Knight et al., 2014, 2017, 2020; May, 2016; Bonial et al., 2020; Bonn et al., 2020), providing a critical source for the statistical machine learning approach to semantic parsing. While these resources are invaluable, producing AMRs is difficult, involving many sub-tasks, such as the annotation of nouns/named-entities, predicate-argument dependencies, co-reference resolution, discourse connectives, negation, and temporal relations. On top of this, annotators would need to be thoroughly trained to navigate a complex annotation tool interface in the process.

Traditional AMR editors typically begin by having the annotator construct a root node and then add additional nodes as children through graph traversal. Nodes are added either through a dashboard made up of a combination of buttons, menus, and entry fields or, through a command-line-like interface with a series of learned commands. Both versions add yet another layer of learning complexity to the already intricate AMR structure. However, this is not simply an AMR problem but a problem for all semantic annotation tasks that involve complex structural layers of annotation.

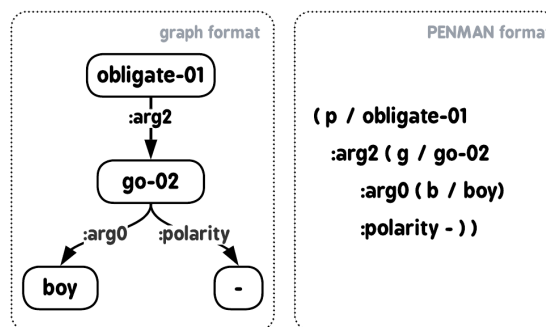


Figure 1: AMR for sentence "The boy must not go." in conventional graph representation format (left) and PENMAN encoding language format (right)

We present an example AMR in Figure 1 for the sentence “The boy must not go.” In an AMR graph, predicates and their corresponding arguments are represented by nodes. In this example, the *go-02* predicate<sup>2</sup> (Palmer et al., 2005; Pradhan et al., 2022) has one argument, which is *boy*. AMR specifies the role of each argument with labeled edges. Core roles, like stereotypical agent, patient, and thematic role, are typically denoted by *arg0*, *arg1*, and *arg2*, respectively. Other non-core roles, which are usually predicate-specific, are directly labeled with their names, such as *location*, *direction*, *time*, and *duration*. AMRs can be expressed in various formats, but graphically annotating their complex structure is impractical. To address this researchers adapted PENMAN notation (Goodman, 2019, 2020), which represents graph structures using bracketing syntax. Labeled edges are encoded with the preceding colon symbol, and opening brackets indicate new AMR nodes. Terminal nodes are denoted by closing parentheses.

Advancements in large language model-based coding assistance, like Codex (Chen et al., 2021) and Copilot by OpenAI and Microsoft have been revolutionizing program synthesis for software engineering tasks. These models are trained for both natural languages and programming languages, enabling them to intelligently complete programs based on code history and human instruction. Drawing inspiration from code-completion approaches, we take a similar path by integrating an AMR parser model alongside a human annotator. This unique combination allows us to streamline the AMR construction process, handling easier yet tedious tasks like named entity sub-graph construction through the parsing assistant. At the same time, more intricate annotation decisions, such as predicate sense distinction, discourse connections, and co-reference resolutions, can be moderated by the annotators themselves, ensuring a balanced and effective approach to AMR annotation.

We summarize our contributions to the semantic annotation task as follows:

- We designed and implemented an innovative online AMR annotation tool that treats semantic annotation as a coding task streamlining the annotation process.
- We present the annotator-centric tool equipped with local Propbank snippet autocomplete and

<sup>2</sup>From PropBank, <https://propbank.github.io/>

full generative model-based suggestions, enhancing the annotation experience for both beginner and experienced annotators.

- We introduce an intuitive click-based matching process for AMR concept alignment, simplifying and accelerating the alignment step for a smoother annotation experience.

Our highly modularized implementation enables easy swapping of language syntax and assistant models, creating a flexible “programming as annotation” paradigm adaptable to various languages and structures.

## 2 Related Work

The two most widely used AMR annotation tools are ISI AMR Editor (ISI-Editor) (Hermjakob, 2013) and Anafora (Chen and Styler, 2013). Both are web-based text annotation tools that focus on different levels of AMR annotation. ISI-Editor is primarily designed for lexical-level AMR annotation, while Anafora is commonly used to construct document-level AMRs based on existing sentence-level AMRs. Our work with CAMRA is primarily comparable to ISI-Editor, as we also focus on sentence-level AMR construction. However, it is worth noting that editing cross-sentence relations, such as inter-sentential coreference resolution, can also be accomplished through CAMRA with relative ease. We will later showcase how to use ISI-Editor in comparison to our approach.

The ISI editor offers comprehensive support for editing an AMR graph through various operators, including *top* to initiate an AMR graph and *add* to create an AMR triplet relation. Figure 2 illustrates an example of this functionality<sup>3</sup>. In the first view, we enter the *add* operator in the command field and submit it to activate the Action template view. Here, we proceed to fill in the new role with the specified head variable, role label, and argument concept node. ISI Editor processes the template form with verification to ensure the action is valid, resulting in an updated AMR displayed in the viewport. In total, annotators have access to 8 core operators that allow them to modify the AMR graph in PENMAN encoding form with shortcuts for advanced users. Additionally, the ISI Editor provides helpful annotation facilitations, such as the ability to

<sup>3</sup>To demonstrate the interaction of the interface in a straightforward manner, we use diagrams in Figure 2 instead of real screenshots. These diagrams faithfully represent the relative positions and interactive logic of the ISI editor.

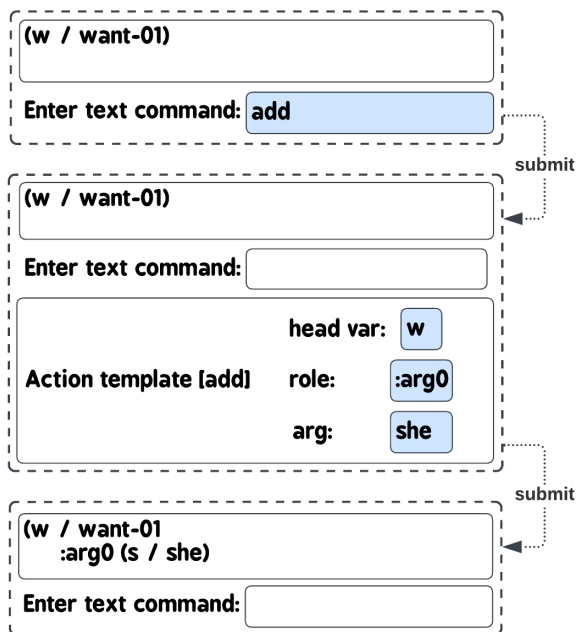


Figure 2: Adding a new argument to predicate node (w / want-01) with ISI editor’s interface. Each dashed line box represents an updated view after submitting the mini form. Blue colored fields of each form represent fields that are required to be filled before submitting

search for existing AMR data and perform error checks on demand. These functionalities are organized within a dashboard interface, equipped with menu buttons that trigger specific features. Using ISI Editor becomes a process of sequentially filling small forms.

There are other annotation tools available for various complex linguistic-driven tasks, such as the UCCAApp (Birch et al., 2016) for Universal Conceptual Cognitive Annotation (Abend and Rapoport, 2013), the brat rapid annotation tool (Stenertorp et al., 2012) for universal dependency tree construction, TreeEditor for Rhetorical Structure Theory (Pajas and Štěpánek, 2008). However, unlike AMR, which can exist without explicit alignment of concept nodes to the natural language surface text, these mentioned tasks are tightly anchored to the surface text. Consequently, they require click-selection-based interactions with the user to initiate the annotation process. Knowtator is another annotation tool that facilitates ontology construction in Protege from text, the UI design of Knowtator (Ogren, 2006) also relies on small form filling.

Furthermore, most syntactic tasks involve a limited number of relationships, typically not exceeding a dozen, in contrast to AMR, where the number

of rolesets directly corresponds to the number of predicates in a given language. In languages like English, the number of predicates can easily surpass 5000. The unique challenge of annotating AMR, coupled with the lack of support for other formalisms, has motivated us to create an annotation tool equipped with a formal language coding environment. This tool aims to enhance the efficiency and accuracy of AMR annotation and provide a novel solution to handle its distinctive complexities.

Recently, significant strides have been made in advancing the development of the model-in-the-loop annotation style, aimed at fostering machine-assisted human annotations. Popular tools, such as Prodigy<sup>4</sup> and INCEpTION (Klie et al., 2018), primarily focus on providing annotation suggestions for Text Classification, Named Entity Recognition, and Entity Relation Extraction. More recent methodologies have expanded these capabilities to encompass entity and event coreference resolution (Bornstein et al., 2020; Ahmed et al., 2023). However, the field of machine-assisted annotations for AMR remains relatively under-explored. Our work endeavors to address this gap, contributing to the enhancement and expansion of this vital aspect of the annotation landscape.

### 3 Design and Features

To enhance the effectiveness of annotator-computer interaction, it is essential for the computer to play an active role in the annotation process, rather than serving merely as a passive typewriter. At the same time, it is important to minimize the need for annotators to frequently shift their attention among different views to memorize local predicate-argument structures temporarily while completing AMR annotations. To address these concerns, we have formulated the following design principles:

- Upon looking up dictionaries like Propbank rolesets or existing annotation examples, the annotator can take advantage of two options. Firstly, they can directly invoke the desired frame within the coding environment, leading to the automatic completion of the target structure. Alternatively, they can easily copy relevant sections from examples and paste them into the coding environment, streamlining the annotation process.

<sup>4</sup>www.prodi.gy

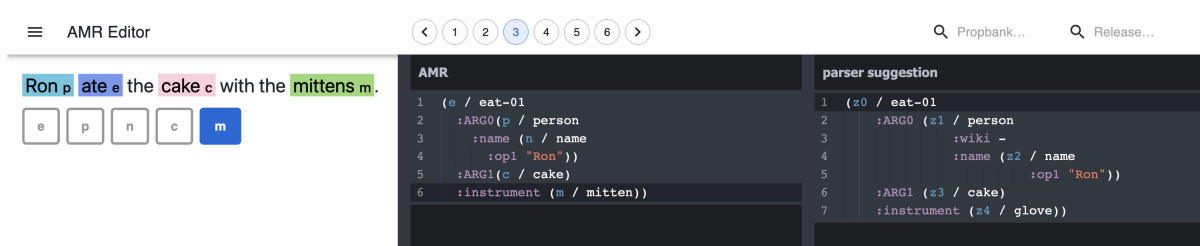


Figure 3: an overview of the CAMRA editor with an annotated example sentence. Left panel is the surface text area with dynamic variable carryover from the constructed AMR code. The middle panel is the main AMR editing area where the string complies with the PENMAN encoding syntax for AMR. The right text panel renders the parser suggestions. Note: this screenshot contains only nonempty part of the UI, the UI is window size responsible.

- The produced AMR should undergo active parsing to ensure its legality and provide valuable feedback to the annotators.
- To make the annotation experience akin to coding, the editor needs to incorporate additional text editing tool features, such as multiple selections, code difference highlight, and editing capabilities, thereby optimizing the annotators' workflow and overall experience.
- The design of the copilot editing environment should be versatile and adaptable to different annotation projects. It should support a general-purpose approach, enabling similar annotation tasks to be accomplished with ease by merely switching formal language syntax as needed.

### 3.1 Features

We show the main app interface in Figure 3.

#### 3.1.1 Annotation panels

CAMRA is primarily composed of three horizontally laid-out panels. The leftmost panel is designed for rendering the surface text and aligning AMR concept nodes to the corresponding text. This panel displays two blocks of information: the surface text itself and the AMR node variable names present in the middle panel.

The middle panel serves as the AMR text editor, equipped with common code editor features, such as syntax highlighting, auto bracket matching and closing, and snippet auto-complete. Writing AMR in this text editor closely resembles writing code in a programming language.

Finally, we utilize the right panel to render the AMR generated by the parser. Annotators have the option to use any part of the parser-suggested AMR by simply copying the text over to the middle panel, facilitating a seamless integration of the parser's

suggestions into the annotator's workflow. This three-panel layout ensures a smooth and intuitive annotation process for CAMRA users.

### 3.2 Autocomplete

CAMRA is equipped with two levels of auto-completion mechanisms: local autocomplete and global autocomplete. The local auto-complete feature considers only the nearest string to provide suggestions for reserved keywords and Propbank templates. This proves useful in cases where AMR relation prompting is required, as it relieves annotators from the burden of remembering every relation precisely. This is particularly helpful for non-core AMR relations, which can be quite lengthy and prone to errors. Additionally, local autocomplete is computationally less intensive compared to global autocomplete, utilizing substring matching as the search algorithm and edit distance as the ranking algorithm when invoked.

Moreover, when snippet autocomplete is activated, the editor holds field-like text spans in memory, allowing annotators to simply type in the value and switch to the next field by pressing the `Tab` key, significantly reducing navigation time within the code.

In contrast, the global autocomplete from the machine learning-based parser considers both the surface text and its generated history, making it more comprehensive than the local autocomplete. However, due to its higher computational cost, the parser suggestion is invoked only once per sentence. We keep the parser suggestion API open to backend updates, enabling the possibility of further tailored parser suggestions that take the users' input into account for even more personalized and refined suggestions.

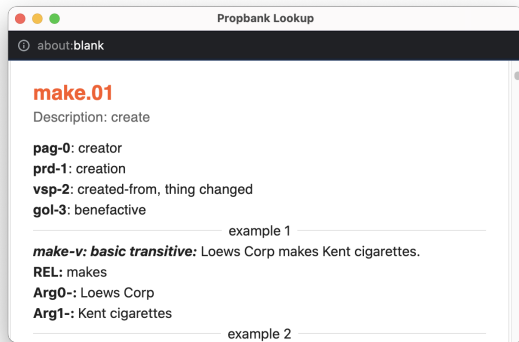


Figure 4: When looking up in the Propbank rolesets for the keyword "make," a persistent new window will appear at the annotator's disposal on the side of the CAMRA's main interface.

### 3.3 Manual Search

CAMRA also incorporates a similar search function for both Propbank and the existing AMR corpus. This feature functions similarly to the ISI editor's search, popping up with a more updated UI design when invoked. In Figure 4 and Figure 5, we demonstrate the search results for the keywords "make" in Propbank and "must" in LDC2020T02 AMR corpus (Knight et al., 2020).

To ensure a clutter-free main annotation window, we have dedicated individual windows to host the search results. These search windows persistently update their content whenever a new search is launched. Annotators can easily browse and perform **Ctrl+F** searches within these windows.

In organizing and highlighting the AMRs in the existing corpus search, we have maintained the same format as the main AMR editing panel. This facilitates straightforward copy-pasting of any desired part of the AMR into the AMR editing panel, providing annotators with direct access to the information they need for a more efficient and streamlined annotation process.

### 3.4 Utility Functions

All managerial and administrative functions are conveniently placed in a hidden menu accessible through the top left corner drawer icon. This menu houses various actions, including uploading a new workset (a text file containing all the target text to be annotated), uploading an annotation checkpoint, profile management, and more. As these functionalities are not the primary focus of our CAMRA and do not represent critical components for this

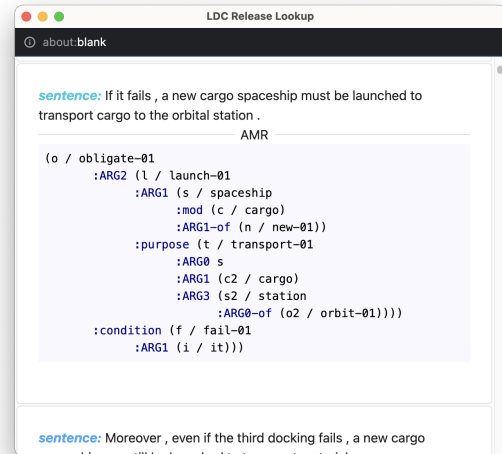


Figure 5: When looking up in the existing AMRs corpora for the keyword "must", another persistent new window will appear at the annotator's disposal on the side of the CAMRA's main interface.

paper, we have opted to exclude them from further discussion but let the reader explore in demonstration.

### 3.5 Language Servers

The core active assistance feature of our annotation tool is powered by language servers on the backend. As elaborated in Section 3.2, we have two layers of language support: a local one and a global one, achieved through two REST-API servers.

Handling managerial tasks such as login, data storage, Propbank and release searching, and parser inquiries is a Django (Django Software Foundation) REST API server's responsibility. In addition, we have set up a separate REST API server dedicated to hosting pre-trained AMR parser models. This division allows for enhanced flexibility in resource distribution. For example, the managerial server can efficiently manage data transactions from the front end without requiring GPU support. On the other hand, most state-of-the-art AMR parser models are large neural network models that greatly benefit from GPU or other accelerating devices' computational power.

The design of CAMRA revolves around modularity as a critical principle, enabling the easy integration of various assistant models. For instance, when annotating unique domains of text, parsers previously trained on different domains may perform poorly, limiting the support they can offer to annotators. By being modular, our system can

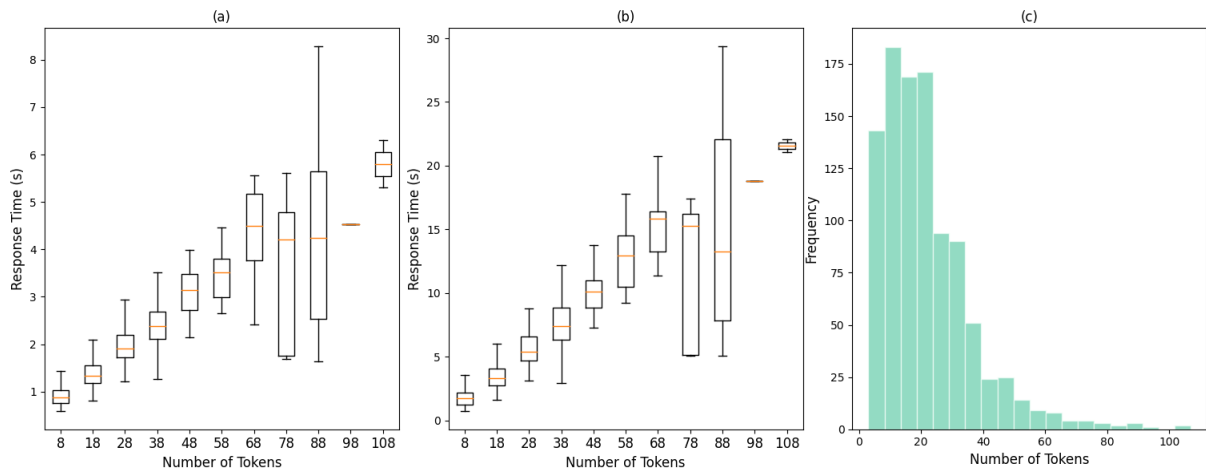


Figure 6: An overview of the AMR parser server’s response time is shown for the same 1000 randomly selected sentences from the LDC2020T02 AMR corpus training set. Figure (a) displays the response time box chart without GPU support, while Figure (b) shows the response time box chart with a single GPU support. Additionally, Figure (c) represents the frequency distribution of sentence lengths among the 1000 sentences. The red middle line of each box candle represents the median, the box specifies the interquartile range (IQR), and the whiskers indicate the 1.5 IQR range.

readily adapt to such scenarios. Additionally, this modularity facilitates the distributed deployment of our system.

At present, we offer support for the SPRING AMR parser (Bevilacqua et al., 2021) trained on LDC and spatial AMR corpus (Bonn et al., 2020) respectively as parsing assistance. However, the flexibility of our design makes it possible to include other assistant models tailored to specific needs in the future. The base model of the SPRING parser is BART-large (Lewis et al., 2020) with nearly 140M parameters and requires approximately 2.2 GB of memory for inference. The integration of the parser facilitates model-in-the-loop learning processes, which can be adapted based on user requirements without significant difficulty.

## 4 Discussion

The primary user experience factor for our annotation tool is the response time of the parser-based language server. To evaluate this response time in relation to sentence length, we conducted tests using 1000 randomly selected sentences from the training set of LDC2020T02. The results are depicted in Figure 6. The average sentence length among these sentences is 21.72 tokens (tokenized by BART tokenizer), with a standard deviation of 14.76. With and without GPU support, the average response times are 1.62 and 4.47 seconds, respectively, with corresponding standard deviations of

0.86 and 3.32 seconds. We present a box chart illustrating the response time distribution in relation to sentence length (token numbers  $n$ ), considering the BART model’s theoretical inference complexity of  $\mathcal{O}(n^2)$  and the prior distribution of sentence lengths. The testing machine has 2.2GHz Intel Xeon (R) CPU (24 cores), 256GB RAM and a NVIDIA Titan Xp GPU (12GB).

## 5 Conclusion and Future Works

CAMRA introduces a novel semantics annotation paradigm with considerable potential for enhancement. Given the similarity of the parser structure, integrating LLM into autocomplete and suggestion output is seamless. We are actively working on fine-tuning LLMs to make parsing copilot suggestions more interactive. To assess the language server’s impact compared to traditional AMR annotation tools, we will conduct a human study involving annotators. Furthermore, we aim to expand the pool of potential annotators, serving the dual purpose of broadening our annotator base and supporting computational semantics education. Collaborating with NLP communities, we plan to extend support to other formalisms and annotation schemes. Moreover, we envision the integration of large language models into the language server, providing more natural language assistance from AI. This advancement could lead to yet another valuable application of large language models, enhancing their inter-

pretability and error resilience through the fusion of neural and symbolic approaches. Such developments offer exciting possibilities for safer and more innovative applications.

## Limitations

CAMRA’s language server support may encounter biases or challenges related to domain shift, as the underlying model’s training data could be skewed towards specific text domains (such as newswire text). This might result in inaccuracies or reduced performance when dealing with text from different domains. Furthermore, while AMR serves as a versatile formalism, our PENMAN syntax design predominantly caters to English, potentially limiting its effectiveness for other languages. Expanding the PENMAN syntax to encompass a wider array of languages would not only improve its cross-linguistic applicability but also enhance the overall usability and inclusivity of the annotation tool.

## Ethics Statement

In addition to the limitations highlighted in the previous section, CAMRA has a core objective of enhancing human-computer communication through UI design and AI assistance. An essential aspect of this endeavor is to ensure that CAMRA users have a comprehensive grasp of how the language server operates and how it impacts annotations. This is achieved through transparent documentation and the provision of mechanisms for understanding the tool’s decision-making process. Furthermore, we place significant emphasis on effective communication with annotators to consider the cultural and domain-specific sensitivities inherent in the text being annotated. Recognizing these nuances is crucial, as any misinterpretation or misrepresentation of cultural contexts could result in erroneous semantic annotations.

## Acknowledgement

This research was supported by the NSF National AI Institute for Student-AI Teaming (iSAT) under grant DRL 2019805. The opinions expressed are those of the authors and do not represent views of the NSF. The authors extend their heartfelt gratitude to Adam Zheng, Brad Johnson, Skatje Myers, Elizabeth Spaulding and Jie Cao for their unwavering support in handling hosting technicalities. Additionally, Dr. Alexis Palmer’s suggestions in shaping the user study design.

## References

- Omri Abend and Ari Rappoport. 2013. [Universal Conceptual Cognitive Annotation \(UCCA\)](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.
- Shafiuddin Rehan Ahmed, Abhijnan Nath, Michael Regan, Adam Pollins, Nikhil Krishnaswamy, and James H. Martin. 2023. [How good is the model in model-in-the-loop event coreference resolution annotation?](#) In *Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII)*, pages 136–145, Toronto, Canada. Association for Computational Linguistics.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.
- Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*.
- Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. [HUME: Human UCCA-based evaluation of machine translation](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1264–1274, Austin, Texas. Association for Computational Linguistics.
- Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. [Dialogue-AMR: Abstract Meaning Representation for dialogue](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.
- Julia Bonn, Martha Palmer, Zheng Cai, and Kristin Wright-Bettner. 2020. [Spatial AMR: Expanded spatial annotation in the context of a grounded Minecraft corpus](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4883–4892, Marseille, France. European Language Resources Association.
- Ari Bornstein, Arie Cattan, and Ido Dagan. 2020. [CoRefi: A crowd sourcing suite for coreference annotation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 205–215, Online. Association for Computational Linguistics.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen

- Krueger, Michael Petrov, Heidy Khlaaf, Girish Sasstry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *CoRR*, abs/2107.03374.
- Wei-Te Chen and Will Styler. 2013. [Anafora: A web-based general purpose annotation tool](#). In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.
- Django Software Foundation. [Django](#).
- Michael Wayne Goodman. 2019. AMR normalization for fairer evaluation. In *Proceedings of the 33rd Pacific Asia Conference on Language, Information, and Computation*, pages 47–56, Hakodate.
- Michael Wayne Goodman. 2020. [Penman: An open-source library and tool for AMR graphs](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.
- Ulf Hermjakob. 2013. [Amr editor: A tool to build abstract meaning representations](#). Marina del Rey, CA. USC Information Sciences Institute.
- 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. Association for Computational Linguistics. Event Title: The 27th International Conference on Computational Linguistics (COLING 2018).
- Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O’Gorman, and Nathan Schneider. 2017. [Abstract meaning representation \(amr\) annotation release 2.0](#). *Linguistic Data Consortium*, LDC2017T10.
- Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O’Gorman, and Nathan Schneider. 2020. [Abstract meaning representation \(amr\) annotation release 3.0](#). *Linguistic Data Consortium*, LDC2020T02.
- Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. [Abstract meaning representation \(amr\) annotation release 1.0](#). *Linguistic Data Consortium*, LDC2014T12.
- 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.
- Jonathan May. 2016. [SemEval-2016 task 8: Meaning representation parsing](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California. Association for Computational Linguistics.
- Philip V. Ogren. 2006. [Knowtator: A protégé plug-in for annotated corpus construction](#). In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Demonstrations*, pages 273–275, New York City, USA. Association for Computational Linguistics.
- Petr Pajas and Jan Štěpánek. 2008. [Recent advances in a feature-rich framework for treebank annotation](#). In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 673–680, Manchester, UK. Coling 2008 Organizing Committee.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.
- Sameer Pradhan, Julia Bonn, Skatje Myers, Kathryn Conger, Tim O’gorman, James Gung, Kristin Wright-bettner, and Martha Palmer. 2022. [PropBank comes of Age—Larger, smarter, and more diverse](#). In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 278–288, Seattle, Washington. 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 Session at EACL 2012*, Avignon, France. Association for Computational Linguistics.



# REACTION MINER: An Integrated System for Chemical Reaction Extraction from Textual Data

Ming Zhong Siru Ouyang Yizhu Jiao Priyanka Kargupta  
Leo Luo Yanzhen Shen Bobby Zhou Xianrui Zhong Xuan Liu  
Hongxiang Li Jinfeng Xiao Minhao Jiang Vivian Hu Xuan Wang  
Heng Ji Martin Burke Huimin Zhao Jiawei Han  
University of Illinois Urbana-Champaign  
{mingz5, siruo2, yizhu2, pk36, hanj}@illinois.edu

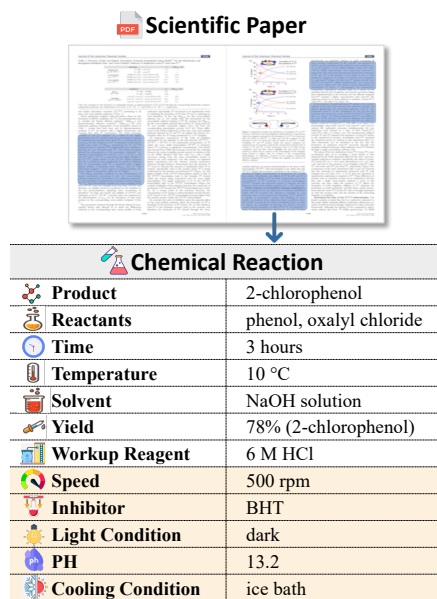
## Abstract

Chemical reactions, as a core entity in the realm of chemistry, hold crucial implications in diverse areas ranging from hands-on laboratory research to advanced computational drug design. Despite a burgeoning interest in employing NLP techniques to extract these reactions, aligning this task with the real-world requirements of chemistry practitioners remains an ongoing challenge. In this paper, we present REACTION MINER, a system specifically designed to interact with raw scientific literature, delivering precise and more informative chemical reactions. Going beyond mere extraction, REACTION MINER integrates a holistic workflow: it accepts PDF files as input, bypassing the need for pre-processing and bolstering user accessibility. Subsequently, a text segmentation module ensures that the refined text encapsulates complete chemical reactions, augmenting the accuracy of extraction. Moreover, REACTION MINER broadens the scope of existing pre-defined reaction roles, including vital attributes previously neglected, thereby offering a more comprehensive depiction of chemical reactions. Evaluations conducted by chemistry domain users highlight the efficacy of each module in our system, demonstrating REACTION MINER as a powerful tool in this field<sup>1</sup>.

## 1 Introduction

Chemical reactions lie at the heart of chemistry, representing the transformative processes that give birth to new substances. The structured format of these reactions paves the way for diverse applications, including synthesis planning (Segler et al., 2018; Genheden et al., 2020), reaction prediction (Schwaller et al., 2018; Coley et al., 2019), and reaction condition recommendation (Gao et al., 2018;

<sup>1</sup>Code, data, and models can be found at: <https://github.com/maszhongming/ReactionMiner>. The link to the video that introduces our system is at: <https://youtu.be/q7P6NWDKcxw>















| Chemical Reaction   |                         |
|---|-------------------------|
|  Product             | 2-chlorophenol          |
|  Reactants           | phenol, oxalyl chloride |
|  Time                | 3 hours                 |
|  Temperature         | 10 °C                   |
|  Solvent            | NaOH solution           |
|  Yield             | 78% (2-chlorophenol)    |
|  Workup Reagent    | 6 M HCl                 |
|  Speed             | 500 rpm                 |
|  Inhibitor         | BHT                     |
|  Light Condition   | dark                    |
|  PH                | 13.2                    |
|  Cooling Condition | ice bath                |

Figure 1: Example from REACTION MINER. Highlighted reaction roles denote the attributes that the previous systems are incapable of extracting.

Maser et al., 2021). In recent years, the combination of chemistry and NLP has emerged as a dynamic research area (Chithrananda et al., 2020; Edwards et al., 2022; Bran et al., 2023), fueled by the prospect of automating the extraction of chemical reactions from vast corpora of scientific papers (Guo et al., 2022; Zhong et al., 2023). By leveraging NLP techniques, researchers can derive crucial insights more rapidly than traditional manual methods (Goodman, 2009), thereby catalyzing progress in myriad chemistry-related domains.

Figure 1 illustrates the goal of this task, which is to mine and extract structured chemical reactions from the extensive chemical literature. Representatively, systems such as OPSIN, CHEMRXNBERT, and REACTIE have emerged to automate the process of chemical reaction extraction, each employing distinct NLP approaches. OPSIN (Lowe, 2012) serves as an early exemplar, utilizing a heuristic-based method that underscores the potential bene-

fits of applying NLP to the realm of chemical data extraction. Subsequently, CHEMRXNBERT (Guo et al., 2022) harnesses the power of pre-training on chemistry literature to foster a deeper understanding of chemical context. REACTIE (Zhong et al., 2023) further refines the Information Extraction (IE) process by reformulating it as a Question Answering (QA) task, facilitating the creation of synthetic data and reducing annotation needs. Despite these significant strides, certain issues persist when these systems are deployed in the hands of real-world chemistry practitioners:

(1) **Input Format Misalignment:** Existing systems are designed to accept plain text as input. However, chemistry practitioners typically engage with literature in PDF format rather than processed text. This disconnect between the format of readily accessible resources and the input requirements introduces a considerable hurdle for practical use.

(2) **Limited Input Granularity:** The typical input for existing systems is confined to a sentence or a fixed-size context window for the extraction. This often leads to a trade-off between over- and under-inclusion of data, with systems either capturing incomplete information about the chemical reaction or introducing irrelevant content.

(3) **Restriction on Extracted Roles:** The current systems focus on extracting pre-defined reaction roles, such as the *reactant*, *product*, *catalyst*, *time*, *temperature*, *yield*, etc. Yet, there are additional attributes that are of considerable interest to practitioners, such as the *experimental procedure* and more nuanced *reaction conditions*, which are commonly overlooked by these systems.

(4) **Output Format Inconsistency:** Lastly, there exists a discrepancy between the output format provided by current systems and what is required by real-world users. Frequently, these systems output incomplete chemical names or incorrect symbols and units, which further complicates the interpretation and application of the extracted information.

To address the outlined challenges, we present REACTION MINER, an integrated system designed to bridge these gaps and cater more closely to the needs of real-world chemistry practitioners. In contrast to existing systems, REACTION MINER incorporates a series of new features:

(1) **PDF-to-Text:** Recognizing that the inherent diversity of templates in chemistry journals presents a formidable challenge for existing tools, we develop a PDF-to-text module specifically tai-

lored for the biochemistry field. It features built-in dynamic similarity calculation functionality based on Sentence-BERT (Reimers and Gurevych, 2019) to alleviate the frequent coherence issues that arise during conversion.

(2) **Text Segmentation:** To ensure the input context includes all necessary details without irrelevant information about chemical reactions, we initially perform text segmentation on the processed text. This involves identifying the central sentence associated with the chemical reaction and subsequently expanding the boundaries of the input text using unsupervised topic segmentation (Choi, 2000).

(3) **Role Enrichment:** To bypass the restrictions imposed by pre-defined label space, we integrate an automatic event mining approach (Jiao et al., 2022) to enrich extracted reaction roles. Furthermore, to enhance our system’s ability to accurately extract newly discovered attributes, we generate synthetic data corresponding to each role based on GPT-4 (OpenAI, 2023) for the training process.

(4) **Unified Reaction Extraction:** Striving to align our system’s output more closely with the requirements of users, we unify the format of existing data and adjust the annotation guideline based on feedback from chemistry practitioners. Concretely, we re-collect (Zhong et al., 2023), re-organize<sup>2</sup>, and re-annotate (Guo et al., 2022) present data, leading to a unified system that caters to the needs of the chemistry community more effectively.

Regarding evaluation, we invite chemistry Ph.D. students to undertake tests and contrast REACTION MINER with current systems. Human evaluation indicates that our system is better aligned with the needs of the chemistry community. Remarkably, even though the architecture of REACTION MINER is built upon LLaMA-7B (Touvron et al., 2023) and LoRA (Hu et al., 2022), it consistently matches or surpasses the performance of large language models across all subtasks. Thus, REACTION MINER represents a step forward in chemical reaction extraction, providing an accessible, high-performing open-source tool to expedite advancements at the intersection of NLP and chemistry.

## 2 Method

In this section, we start with the task formulation to provide an overarching perspective of REACTION MINER, subsequently delving into each module.

<sup>2</sup>Data from <https://docs.open-reaction-database.org/en/latest>.

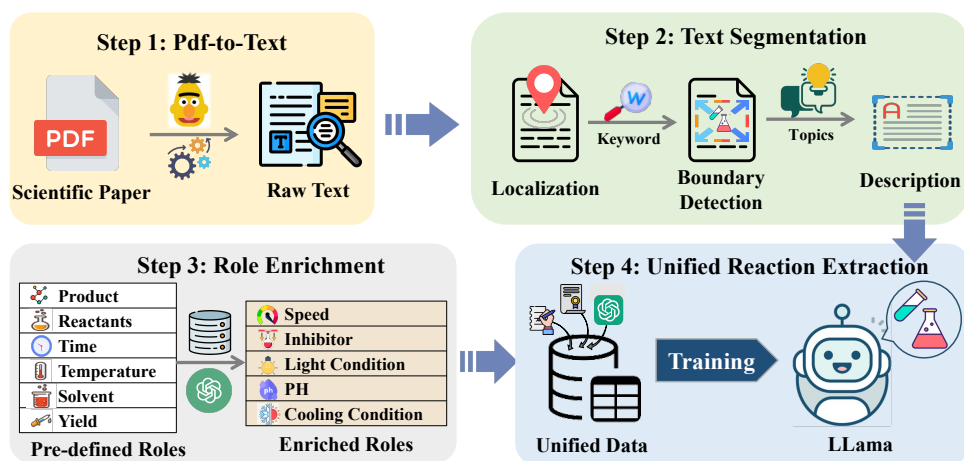


Figure 2: Overall framework for REACTION MINER.

## 2.1 Task Formulation

Given any PDF file that contains chemical context, the PDF-to-text module initially converts it into processed text, represented as  $T$ . This is followed by the segmentation model identifying and segmenting  $T$  into  $k$  relevant passages  $\mathcal{P} = \{P_1, \dots, P_k\}$  associated with the chemical reaction to serve as inputs for reaction extraction. For each passage  $P$ , the objective is to extract all the structured chemical reactions  $\mathcal{C}$  present within  $P$ , where every reaction  $C \in \mathcal{C}$  comprises  $n$  role-argument pairs  $\{(r_1, a_1), \dots, (r_n, a_n)\}$ . Here, the term “role”  $r$  refers to a crucial attribute of a chemical reaction, such as the *product*, *reactant*, *catalyst*, *solvent*, *time*, *temperature*, *yield*, etc., while the corresponding “argument”  $a$  is the extracted span of the corresponding reaction role in the input  $P$ .

## 2.2 PDF-to-Text

The common format in which practitioners access literature is PDF rather than processed text, making it a more appropriate input for an extraction system. However, the inherent diversity in templates used by various chemical journals presents a significant challenge for the development of reliable PDF conversion tools. Current popular methods, such as Gorbid<sup>3</sup>, which is used in S2ORC (Lo et al., 2020), and SymbolScrapper<sup>4</sup>, either overlook short paragraphs or pose incoherence problems (such as intermixing header, footer, or caption information with the body text). These issues can ultimately impact the performance of subsequent extraction.

To address these challenges, we devise our own PDF-to-Text parser. It operates in three stages: 1)

<sup>3</sup><https://github.com/kermitt2/grobid>

<sup>4</sup><https://github.com/zanibbi/SymbolScrapper>

converting the given PDF files into XML format via SymbolScrapper, 2) parsing the XML file into content paragraphs while excluding figures, tables, and captions from the body text through regular expressions, and 3) filtering out incoherent and irrelevant information (e.g., headers, footers, and references). This last step leverages the representative power of a pre-trained language model: we dynamically maintain a set of paragraphs representing the main content of the preceding paragraph and use the average embedding obtained from Sentence-BERT (Reimers and Gurevych, 2019) as the current anchor embedding. Subsequent paragraphs are filtered out if their cosine similarity to the anchor embedding falls below a certain threshold.

## 2.3 Text Segmentation

Text segmentation aims to segment out the reaction-related context from an entire paper. Segmenting the text into more manageable units enables the precise and efficient identification of reaction components. Its process is comprised of two main steps:

### Keyword-based central sentence localization.

The first step in the process involves leveraging the fundamental definition of chemical reactions, with the underlying hypothesis that all chemical reactions involve specific products (Muller, 1994). Human readers often intuitively identify products within textual information through linguistic cues, such as specific keywords. Consider the sentence, “Removal of the TBS moiety of 17 was carried out with TBAF/AcOH in MeCN at 60 °C to give diol 20 in 86% yield.” From the presence of the word “yield”, one can deduce that the product is “diol 20”. Building on this insight, we curate a set of 35 keywords that are demonstrative of products in chem-

ical reactions. When scanning through the text, sentences containing these keywords are identified as central to understanding the reaction context.

**Topic-aided boundary detection.** Once the central sentence is identified, the subsequent step involves detecting the contextual boundary related to the specific chemical reaction. Research has shown that the integration of semantic information through topic models substantially enhances the effectiveness of segmentation algorithms (Riedl and Biemann, 2012; Alemi and Ginsparg, 2015). Motivated by these findings, we employ semantic topical information within the texts to more accurately discern different context blocks associated with various chemical reactions. Specifically, C99 (Choi, 2000), a widely-recognized method for topical detection is used. It annotates sentences with matching tags if they pertain to the same topical group. Any topical group containing the identified central sentence is considered as a segmented context, relevant to particular chemical reactions.

## 2.4 Role Enrichment

Typically, prevalent systems can extract 9 reaction roles, namely *product*, *reactant*, *catalyst*, *solvent*, *workup reagent*, *reaction type*, *time*, *temperature*, and *yield*. However, this coverage is insufficient for capturing all vital properties of a chemical reaction. To address this, we apply an event mining approach (Jiao et al., 2022) to the chemistry literature. It is grounded in the identification of all entities within a text, and then allowing T5 (Raffel et al., 2020) to generate the corresponding entity type to discover frequent new reaction roles. Upon manual review and filtering by chemistry practitioners, we integrate an additional 10 new reaction roles, with the complete list available in the Appendix C.

Simultaneously, an obstacle arises with the enrichment of reaction roles due to the current scarcity of suitable training data. To tackle this issue, we annotate descriptions of the newly added reaction roles and provide three demonstrations. This enables in-context learning, allowing GPT-4 to generate chemical text, alongside the corresponding extracted chemical reactions. We then institute a filtering process where we: 1) eliminate samples where the generated argument does not exist in the original text, 2) remove the generated roles does not appear in the label space, and 3) in tandem with REACTIE (Zhong et al., 2023), remove samples where the generated products exhibit low proba-

bilities of extraction in REACTIE. As a result, we manage to enrich the spectrum of new roles that need extraction for the chemical reaction extraction task, coupled with the associated training data.

## 2.5 Reaction Extraction

Despite the existence of datasets for the chemical reaction extraction task, they vary in terms of reaction roles, output formats, and annotation guidelines. This underscores the need for a standardized and unified data format, which stands as a critical prerequisite for the development of a universally applicable system. Accordingly, we establish this requisite uniformity by re-collecting, re-organizing, and re-annotating the existing data as follows.

**Re-collecting Negative Samples.** While a keyword-based approach in the segmentation module currently serves to locate chemical reactions, this method is high in recall but low in precision. That is, passages containing the designated keywords do not necessarily describe chemical reactions. Thus, we incorporate negative samples — instances where the input text does not contain chemical reactions — into the reaction extraction training. For these samples, models should output “No complete chemical reaction”. We achieve this by running our segmentation model on the chemistry literature and re-collecting the filtered segments as negative samples.

**Re-organizing Open Reaction Database.** The Open Reaction Database<sup>5</sup>, a publicly available repository of chemical reactions, primarily contains data from patent literature, with ground truths extracted via the rule-based system OPSIN (Lowe, 2012, 2018). We re-organize the data format within this database, filtering out samples with semantically repetitive content in the input text. Additionally, our sampling procedure prioritizes scientific papers and examples containing multiple chemical reactions, forming part of our final training data.

**Re-annotating Reaction Corpus.** Although the Reaction Corpus (Guo et al., 2022) is a manually annotated dataset, its annotation guideline prompts the output chemical to be represented as a unique token of compound rather than the full name, reducing user readability. Moreover, its output format occasionally contains incorrect symbols and units due to tokenization errors. Thus, we re-annotate the

<sup>5</sup><https://docs.open-reaction-database.org>

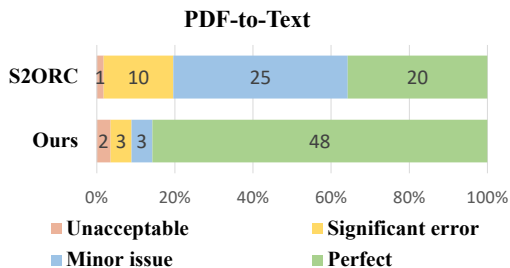


Figure 3: Human evaluation for PDF-to-Text.

training set to eliminate such minor inconsistencies, thereby achieving a unified data format.

Utilizing the above data, we train the LLaMA-7b (Touvron et al., 2023) in a parameter-efficient manner using the LoRA method (Hu et al., 2022), enabling it to function as a chemical reaction extractor within our REACTION MINER framework. More details can be found in the Appendix D.

### 3 Experiments

In this section, we evaluate the performance of REACTION MINER by testing its three core modules.

#### 3.1 PDF-to-Text

**Experimental Setup.** To evaluate the quality and generalization of our PDF-to-Text parser, we randomly sample 56 papers from the top 10 most influential chemical journals across various scholarly publishers (i.e., The Royal Society of Chemistry and American Chemistry Society). For each sample, we manually compare the resulting text with the original PDF using a four-level rating system: perfect, minor issue, significant error, and unacceptable. Here, “minor issue” indicates a few incoherent lines, whereas “significant error” refers to an omission or mixture of several paragraphs, significantly impacting readability.

**Results.** The results of the human evaluation are shown in Figure 3. S2ORC (Lo et al., 2020) is a vast corpus of 81.1M English-language academic papers, thus its PDF-to-Text tool is widely employed. However, given the wide variety of journal templates in the chemical literature, it achieves a “perfect” rating in 20 instances, implying it only completely preserves 35.7% of the original text from the PDFs. Moreover, it frequently overlooks paragraphs or includes unrelated content. Conversely, the PDF-to-Text component in REACTION MINER flawlessly processes the text in 85.7% of the instances, underscoring the effectiveness of our

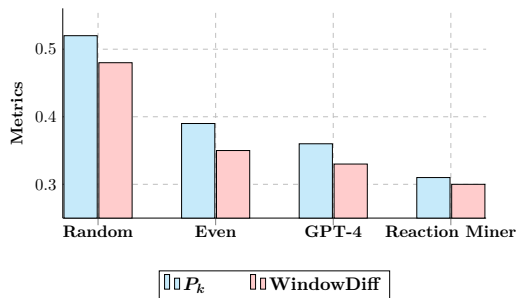


Figure 4: Evaluation results for text segmentation. Lower values indicate better performance.

proposed dynamic similarity computation component in resolving the incoherence issue.

#### 3.2 Text Segmentation

**Experimental Setup.** We randomly collect 50 samples from chemical literature and employ 3 graduate students with chemical backgrounds to annotate the reaction-related context from the text. The average length of samples is 328.26, resembling a short article, and the average number of segments is 2.34 per sample. Three segmentation baselines are used for comparison: 1) Random: segment boundaries are randomly assigned; 2) Even: segment boundaries are evenly placed every  $k$  sentences; 3) GPT-4: employ GPT-4 to identify sentences related to chemical reactions. Two common measures for text segmentation,  $P_k$  (Beeferman et al., 1999) and WindowDiff (Pevzner and Hearst, 2002) are leveraged as evaluation metrics, with lower values indicating better performance. During experiments, we set the size of the sliding window in WindowDiff  $k = 2$ , equaling  $k$  in Even baseline.

**Results.** Figure 4 presents a comprehensive summary of the text segmentation results obtained using the constructed test set. The analysis reveals that our proposed method substantially surpasses all the existing baseline methods with respect to both  $P_k$  and WindowDiff metrics, underlining its ability to accurately identify segment boundaries. A particular highlight of our findings is the superiority of our approach, REACTION MINER, over the strongest proprietary model, GPT-4, by improvements of 16.2% ( $0.37 \Rightarrow 0.31$ ) and 9.1% ( $0.33 \Rightarrow 0.30$ ) for  $P_k$  and WindowDiff, respectively. This not only establishes the efficacy of our method but also provides critical insights into the existing limitations of GPT-4, one of the most sophisticated language models to date, in the realm of text segmentation. Interestingly, GPT-4’s performance is found to be comparable to the Even method, point-

| <b>Input Text:</b> It is worth mentioning that when the reaction of 2-cyano-N,N-dimethylethanethioamide (1s) with 3-phenylpropionaldehyde (2a) was carried out at room temperature under nitrogen for 5 h, an aldol condensation product 5a was isolated in 73% yield, which can be further converted to 2-aminothienyl ether 3sa in 85% yield under the optimized reaction conditions. |   |  |
|---|---|--|
| <b>ReactIE (Significant Error)</b>  | <b>GPT-4 (Minor Issue)</b>                  | <b>Reaction Miner (Perfect)</b>        |
| Reaction 1:   | Reaction 1                                  | Reaction 1                             |
| Product: 5a   | Product: aldol condensation product 5a      | Product: 5a                            |
| Reactants: (1s), (2a), ...  | Reactant: 2-cyano-... (1s)                  | Reactant: 2-cyano-... (1s), 3-... (2a) |
| Reaction type: aldol condensation   | Reactant: 3-... (2a)                        | Atmosphere: nitrogen                   |
| Catalyst: nitrogen  | Atmosphere: nitrogen                        | Reaction type: aldol condensation      |
| Temperature: room   | Temperature: room temperature               | Temperature: room temperature          |
| Time: 5 h   | Time: 5 h                                   | Time: 5 h                              |
| Yield: 73%  | Yield: 73% (aldol condensation product 5a); | Yield: 73% (5a)                        |
| Reaction 2:   | Reaction 2                                  | Reaction 2                             |
| Product: 3sa  | Product: 2-aminothienyl ether 3sa           | Product: 2-aminothienyl ether 3sa      |
| Reactants: (1s), (2a), ...  | Reaction type: optimized reaction           | Reactant: 5a                           |
| Yield: 85%  | Yield: 85% (2-aminothienyl ether 3sa)       | Yield: 85% (2-aminothienyl ether 3sa)  |

Figure 5: Case study for the different reaction extraction systems. The blue text indicates the results of the human evaluation, and the red text represents the parts of the model output that are problematic. In addition, the output of GPT-4 misses a few reaction roles, such as the “Reactant” in Reaction 2.

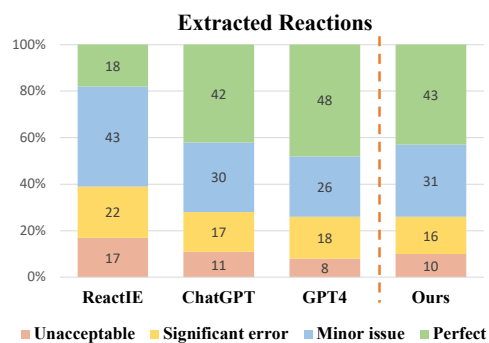


Figure 6: Human evaluation for reaction extraction.

ing to specific areas where even such a powerful model may exhibit weaknesses. Details of model outputs and implementation are in Appendix B.

### 3.3 Reaction Extraction

**Experimental Setup.** To exclusively assess the performance of the reaction extraction module, we curate a test set by manually annotating 100 samples that encompass the complete segment. We evaluate and contrast four distinct systems: 1) *ReactIE* (Zhong et al., 2023), which stands as the state-of-the-art chemical reaction extraction system, built upon Flan-T5 (Chung et al., 2022); 2) *ChatGPT*, a proprietary model for conversational scenarios based on InstructGPT (Ouyang et al., 2022); 3) *GPT-4* (OpenAI, 2023): the most advanced proprietary model accessible at present, and 4) Reaction extraction module in our *REACTION MINER* that utilizes LLaMA-7b as the backbone. Evaluation details are provided in Appendix D.

**Results.** The evaluation results are detailed in Figure 6. Despite being the previous best extrac-

tion system, *ReactIE* only perfectly matches the user’s needs in 18% of the cases, with frequent minor issues and significant errors. Such shortcomings can be attributed to frequent formatting inconsistencies and the omission of certain reaction roles inherent in its prior data format. Contrastingly, the performance of *REACTION MINER* aligns more closely with *ChatGPT* and *GPT-4*, yielding a satisfactory outcome (“perfect” and “minor issue”) in approximately 75% of cases. A salient point to highlight is that although having a considerably smaller parameter set than its proprietary counterparts and being open-source, *REACTION MINER* offers a performance that is on par. This positions it as a remarkably efficient open-source tool in this field. Figure 5 provides a more granular view of the outputs from different systems. In the given exemplar, *ReactIE* inaccurately identifies reactants and catalysts, resulting in it being categorized under “significant error”. *GPT-4*, on the other hand, encounters a few formatting challenges, and misses reactants in the second reaction. In contrast, *REACTION MINER* adeptly extracts all the pertinent reaction roles, facilitating a comprehensive understanding of the given chemical reaction.

## 4 Conclusion

In our exploration, we present *REACTION MINER*, an integrated system adept at extracting chemical reactions directly from raw scientific PDFs. Beyond mere extraction, it offers enhanced accuracy by broadening the scope of reaction roles and eliminating prior gaps. Feedback from chemistry experts marks it as a powerful tool for the field.

## Acknowledgements

We would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported by the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

## References

- Alexander A. Alemi and Paul Ginsparg. 2015. [Text segmentation based on semantic word embeddings](#). *CoRR*, abs/1503.05543.
- Doug Beeferman, Adam L. Berger, and John D. Lafferty. 1999. [Statistical models for text segmentation](#). *Mach. Learn.*, 34(1-3):177–210.
- Andres M Bran, Sam Cox, Andrew D White, and Philippe Schwaller. 2023. [Chemcrow: Augmenting large-language models with chemistry tools](#). *arXiv preprint arXiv:2304.05376*.
- Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. [Chemberta: Large-scale self-supervised pretraining for molecular property prediction](#). *CoRR*, abs/2010.09885.
- Freddy Y. Y. Choi. 2000. [Advances in domain independent linear text segmentation](#). In *6th Applied Natural Language Processing Conference, ANLP 2000, Seattle, Washington, USA, April 29 - May 4, 2000*, pages 26–33. ACL.
- 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.
- Connor W Coley, Wengong Jin, Luke Rogers, Timothy F Jamison, Tommi S Jaakkola, William H Green, Regina Barzilay, and Klavs F Jensen. 2019. [A graph-convolutional neural network model for the prediction of chemical reactivity](#). *Chemical science*, 10(2):370–377.
- Carl Edwards, Tuan Manh Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. [Translation between molecules and natural language](#). 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 375–413. Association for Computational Linguistics.
- Hanyu Gao, Thomas J Struble, Connor W Coley, Yuran Wang, William H Green, and Klavs F Jensen. 2018. [Using machine learning to predict suitable conditions for organic reactions](#). *ACS central science*, 4(11):1465–1476.
- Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Jannik Bjerrum. 2020. [Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning](#). *J. Cheminformatics*, 12(1):70.
- Jonathan M. Goodman. 2009. [Computer software review: Reaxys](#). *J. Chem. Inf. Model.*, 49(12):2897–2898.
- Jiang Guo, A. Santiago Ibanez-Lopez, Hanyu Gao, Victor Quach, Connor W. Coley, Klavs F. Jensen, and Regina Barzilay. 2022. [Automated chemical reaction extraction from scientific literature](#). *J. Chem. Inf. Model.*, 62(9):2035–2045.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Yizhu Jiao, Sha Li, Yiqing Xie, Ming Zhong, Heng Ji, and Jiawei Han. 2022. [Open-vocabulary argument role prediction for event extraction](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5404–5418. Association for Computational Linguistics.
- Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020. [S2ORC: the semantic scholar open research corpus](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4969–4983. Association for Computational Linguistics.
- Daniel Lowe. 2018. [Chemical reactions from us patents \(1976-sep2016\)](#). *doi*, 10:m9.
- Daniel M. Lowe. 2012. [Extraction of chemical structures and reactions from the literature](#). Ph.D. thesis, University of Cambridge, UK.
- Michael R Maser, Alexander Y Cui, Serim Ryou, Travis J DeLano, Yisong Yue, and Sarah E Reisman. 2021. [Multilabel classification models for the prediction of cross-coupling reaction conditions](#). *Journal of Chemical Information and Modeling*, 61(1):156–166.
- P. Muller. 1994. [Glossary of terms used in physical organic chemistry \(iupac recommendations 1994\)](#). *Pure and Applied Chemistry*, 66(5):1077–1184.

- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- 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*.
- Lev Pevzner and Marti A. Hearst. 2002. [A critique and improvement of an evaluation metric for text segmentation](#). *Comput. Linguistics*, 28(1):19–36.
- 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.
- 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.
- Martin Riedl and Chris Biemann. 2012. [How text segmentation algorithms gain from topic models](#). In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 553–557, Montréal, Canada. Association for Computational Linguistics.
- Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. 2018. [“found in translation”: predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models](#). *Chemical science*, 9(28):6091–6098.
- Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. 2018. [Planning chemical syntheses with deep neural networks and symbolic AI](#). *Nat.*, 555(7698):604–610.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#). *CoRR*, abs/2302.13971.
- Ming Zhong, Siru Ouyang, Minhao Jiang, Vivian Hu, Yizhu Jiao, Xuan Wang, and Jiawei Han. 2023. [Reactie: Enhancing chemical reaction extraction with weak supervision](#). In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12120–12130. Association for Computational Linguistics.



## A Details for PDF-to-Text

**Implementation Details.** To filter out incoherent information, we dynamically maintain a set of anchor paragraphs. To avoid anchoring in the reference section, of which the embedding is totally different from other sections of a paper, we deliberately select the longest paragraph in the first one-third of the paper as the first anchor paragraph. To obtain embedding for each paragraph, we adopt a pre-trained sentence-transformer all-mpnet-base-v2<sup>6</sup>. Then we iterate through each paragraph, computing an average cosine similarity score between the paragraph embedding and each anchor paragraph. If the cosine similarity score falls below a threshold 0.12, we drop the content, otherwise, we add the current paragraph into anchor paragraphs. If the number of anchor paragraphs is more than 5, we pop the front-most anchor paragraph.

**Evaluation Details.** Papers used in evaluation are sampled from the following journals: Journal of American Chemistry Society, Angewandte Chemie International Edition, Chemical Communication, Chemical Society Reviews, Organic Letters, ACS Catalysis, The Journal of Organic Chemistry, Chemical Science, Organic & Biomolecular Chemistry, and Accounts of Chemical Research.

## B Details for Text Segmentation

In this section, we provide supplementary information on the text segmentation module.

**Keywords curation.** All the following words are used as keywords when locating central sentences: { 'yields', 'yielded', 'yield', 'yielding', 'afforded', 'afford', 'affording', 'affords', 'produce', 'produces', 'produced', 'producing', 'obtained', 'obtain', 'obtaining', 'obtains', 'transformed', 'transform', 'transforms', 'transforming', 'convert', 'conversion', 'converted', 'converts', 'converting', 'synthesize', 'synthesized', 'synthesis', 'desired', 'desiring' } Note that different words could have different forms of the same meaning.

**Case Study.** Figure 7 demonstrates the segmentation results from different models for one specific example in the test set. We can see that boundary relations predicted by GPT-4 are relatively simple compared to REACTION MINER, revealing its

<sup>6</sup><https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

shortcomings in identifying contexts that are related to chemical reactions. REACTION MINER, on the other hand, provides a much similar boundary detection compared with the ground truth annotations.

## C Details for Role Enrichment

**Reaction Role Definitions.** Here we provide a complete set of 19 reaction roles as well as their definitions, including 9 roles prevalent in existing systems and another 10 roles enriched.

The following are the 9 reaction roles that are used in most existing systems:

- (1) **Product:** Chemical substance that is the final outcome (major product) of the reaction.
- (2) **Reactant:** Chemical substances that contribute heavy atoms to the product.
- (3) **Catalyst:** Chemical substances that participate in the reaction but do not contribute heavy atoms (e.g., acid, base, metal complexes).
- (4) **Workup reagents:** Chemical substances that are used after the reactions to terminate the reactions or obtain the products (e.g., quenching reagents, extraction solvent, neutralizing acids/bases).
- (5) **Solvent:** Chemical substances that are used to dissolve/mix other chemicals, typically quantified by volume and used in superstoichiometric amounts (e.g., water, toluene, THF).
- (6) **Time:** Duration of the reaction performed.
- (7) **Yield:** Yield of the product.
- (8) **Reaction type:** Descriptions about the type of chemical reaction.
- (9) **Temperature:** Temperature at which the reaction occurs.

To capture sufficient information on chemical reactions and ensure coverage, we enrich the existing roles and obtain another 10 reaction roles mined from the chemistry literature. The definitions of enriched 10 reaction roles are listed below:

- (1) **Atmosphere:** The type of gas present during the reaction can be crucial, especially for reactions sensitive to oxygen or moisture (e.g., reactions carried out under nitrogen or argon atmosphere).
- (2) **Inhibitor:** Chemical substances introduced into the reaction environment to slow down, or completely halt, the reaction (e.g., a radical inhibitor like butylated hydroxytoluene (BHT) in polymerization reactions, a catalyst poison like sulfur in Haber process).

---

## Ground Truth

---

When 0.5 equiv of Na<sub>2</sub>S<sub>2</sub>O<sub>8</sub> was combined with 2 equiv of Selectfluor and 20 mol % AgNO<sub>3</sub>, the reaction time decreased from 2 h to 15 min for all substrates, forming alkyl fluorides in excellent yields. The significant acceleration of rate in the decarboxylative fluorination has led to a more efficient process, suggesting that this approach may be useful for 18F labeling. In the seminal work of Li, a Ag(II) fluoride is proposed as the active fluorine atom source in the reaction as opposed to Selectfluor. To support this supposition, Li and co-workers heated the combination of tert-butyl-2-ethyltetradecaneperox-oate and Selectfluor in a sealed tube to 120 °C for 2 h. When the reaction was run in acetone, a 22% yield of 3-fluoropentadecane was obtained, whereas when the reaction was run in 50:50 acetone/water, only a 4% of fluorinated product was obtained. On the basis of these findings, Li proposed that fluorine atom transfer from Selectfluor to alkyl radicals is unlikely to be involved in the Ag-catalyzed process. Selectfluor is reported to be unstable in water at high temperature, forming HF through reaction of the reagent and water. To examine this, we heated Selectfluor in acetone-d<sub>6</sub>/D<sub>2</sub>O to 120 °C in a sealed tube for 2 h. After cooling to room temperature, a sample was removed and examined by 1H NMR, showing that 80% of the reagent decomposed to the defluorinated chloromethyl derivative (see experiment were likely not conducive to testing whether radicals can abstract a fluorine atom from Selectfluor). In addition, if a Ag(II)-F intermediate was formed during the reaction, it can be present only in a catalytic amount (at most). During its formation, radicals are also generated in a catalytic amount, so the likelihood of a small amount of radical being fluorinated by a small amount of Ag(II)-F in the presence of excess Selectfluor is unlikely. Finally, there is a large body of evidence showing that Selectfluor and similar electrophilic fluorinating reagents react with radicals to form C-F bonds.

---

## GPT-4 Segmentation

---

When 0.5 equiv of Na<sub>2</sub>S<sub>2</sub>O<sub>8</sub> was combined with 2 equiv of Selectfluor and 20 mol % AgNO<sub>3</sub>, the reaction time decreased from 2 h to 15 min for all substrates, forming alkyl fluorides in excellent yields. The significant acceleration of rate in the decarboxylative fluorination has led to a more efficient process, suggesting that this approach may be useful for 18F labeling. In the seminal work of Li, a Ag(II) fluoride is proposed as the active fluorine atom source in the reaction as opposed to Selectfluor. To support this supposition, Li and co-workers heated the combination of tert-butyl-2-ethyltetradecaneperox-oate and Selectfluor in a sealed tube to 120 °C for 2 h. When the reaction was run in acetone, a 22% yield of 3-fluoropentadecane was obtained, whereas when the reaction was run in 50:50 acetone/water, only a 4% of fluorinated product was obtained. On the basis of these findings, Li proposed that fluorine atom transfer from Selectfluor to alkyl radicals is unlikely to be involved in the Ag-catalyzed process. Selectfluor is reported to be unstable in water at high temperature, forming HF through reaction of the reagent and water. To examine this, we heated Selectfluor in acetone-d<sub>6</sub>/D<sub>2</sub>O to 120 °C in a sealed tube for 2 h. After cooling to room temperature, a sample was removed and examined by 1H NMR, showing that 80% of the reagent decomposed to the defluorinated chloromethyl derivative (see experiment were likely not conducive to testing whether radicals can abstract a fluorine atom from Selectfluor). In addition, if a Ag(II)-F intermediate was formed during the reaction, it can be present only in a catalytic amount (at most). During its formation, radicals are also generated in a catalytic amount, so the likelihood of a small amount of radical being fluorinated by a small amount of Ag(II)-F in the presence of excess Selectfluor is unlikely. Finally, there is a large body of evidence showing that Selectfluor and similar electrophilic fluorinating reagents react with radicals to form C-F bonds.

---

## Reaction Miner

---

When 0.5 equiv of Na<sub>2</sub>S<sub>2</sub>O<sub>8</sub> was combined with 2 equiv of Selectfluor and 20 mol % AgNO<sub>3</sub>, the reaction time decreased from 2 h to 15 min for all substrates, forming alkyl fluorides in excellent yields. The significant acceleration of rate in the decarboxylative fluorination has led to a more efficient process, suggesting that this approach may be useful for 18F labeling. In the seminal work of Li, a Ag(II) fluoride is proposed as the active fluorine atom source in the reaction as opposed to Selectfluor. To support this supposition, Li and co-workers heated the combination of tert-butyl-2-ethyltetradecaneperox-oate and Selectfluor in a sealed tube to 120 °C for 2 h. When the reaction was run in acetone, a 22% yield of 3-fluoropentadecane was obtained, whereas when the reaction was run in 50:50 acetone/water, only a 4% of fluorinated product was obtained. On the basis of these findings, Li proposed that fluorine atom transfer from Selectfluor to alkyl radicals is unlikely to be involved in the Ag-catalyzed process. Selectfluor is reported to be unstable in water at high temperature, forming HF through reaction of the reagent and water. To examine this, we heated Selectfluor in acetone-d<sub>6</sub>/D<sub>2</sub>O to 120 °C in a sealed tube for 2 h. After cooling to room temperature, a sample was removed and examined by 1H NMR, showing that 80% of the reagent decomposed to the defluorinated chloromethyl derivative (see experiment were likely not conducive to testing whether radicals can abstract a fluorine atom from Selectfluor). In addition, if a Ag(II)-F intermediate was formed during the reaction, it can be present only in a catalytic amount (at most). During its formation, radicals are also generated in a catalytic amount, so the likelihood of a small amount of radical being fluorinated by a small amount of Ag(II)-F in the presence of excess Selectfluor is unlikely. Finally, there is a large body of evidence showing that Selectfluor and similar electrophilic fluorinating reagents react with radicals to form C-F bonds.

---

Figure 7: Case study for text segmentation conducted by the two most superior models. Note that the gray highlights are the boundary sentences.

- (3) Pressure: The pressure at which the reaction is carried out, which may be above or below atmospheric pressure, depending on the requirements of the reaction.
- (4) pH: If the reaction is carried out in an aqueous solution, the pH of the solution could be an important factor.
- (5) Speed: Some reactions require specific stirring or mixing speeds, which can significantly impact the outcome of the reaction.
- (6) Vacuum condition: Some reactions or post-reaction procedures (like solvent evaporation) require specific vacuum conditions to proceed effectively.
- (7) Light condition: Certain reactions (photochemical reactions) require specific light conditions - wavelengths, intensity, or duration - to proceed.
- (8) Cooling/Heating Condition: The specific conditions under which a reaction mixture is heated or cooled, including the temperature range, the rate of temperature change, and the duration at each temperature.
- (9) Spectroscopic data: Information collected about the product using various spectroscopic methods such as NMR, IR, MS, which can help confirm its structure and composition.
- (10) Procedure: The specific steps followed in conducting the reaction, including the order of addition of reactants, the sequence of reactions in multi-step syntheses, etc.

| Source                 | # Texts | # Reactions |
|------------------------|---------|-------------|
| Open Reaction Database | 200,000 | 201,666     |
| GPT-4                  | 35,173  | 48,305      |
| Chemistry Literature   | 25,000  | 0           |
| Reaction Corpus        | 385     | 491         |
| Total                  | 260,558 | 250,462     |

Table 1: Data statistics for reaction extraction. A text can contain more than one chemical reaction. “Chemistry Literature” is used as negative samples, i.e. the input text does not contain a chemical reaction.

**Training Data Generation.** To pair the newly enriched reaction roles with corresponding training data, we leverage GPT-4 with in-context learning technique for data generation. The detailed prompts are shown in Figure 8.

## D Details for Reaction Extraction

**Implementation Details.** By re-collecting negative samples, re-organizing open reaction database, and re-annotating the reaction corpus, we gather the data statistics as presented in Table 1.

For the model training, we adopt the parameter-efficient approach LoRA to train the LLaMA-7b model as it is more computationally efficient and yields similar performance to full finetuning. The training is divided into two phases: we first conduct a two-epoch training on the reorganized open reaction database data and 22,000 negative samples from chemistry literature, expecting that the model can learn the preliminary chemistry knowledge in the first phase. Then, we perform the second stage of finetuning for a total of 20 epochs on the GPT-4 generated data, re-annotated reaction corpus, and 3,000 negative samples, aiming to allow the model to further learn to extract enriched reaction roles. For both phases of training, the batch size is set to 128, as well as the learning rate is  $3e-4$  with a warm-up ratio of 0.03.

**Evaluation Details.** For models belonging to the GPT class, we precede the input text with a comprehensive task definition and three exemplar demonstrations. Two Ph.D. students majoring in chemistry are invited to participate in manually evaluating the outputs of each model. The evaluation standards are consistently categorized into four tiers: perfect, minor issue, significant error, and irrelevant or invalid. The evaluation criteria continue to be divided into four levels: “perfect”, “minor issue”, “significant error” and “unacceptable”.

The human evaluation system is built on an open-source annotation tool, doccano<sup>7</sup>. The evaluators coordinate the standards for the rating system before starting annotation and then each of them rate all the instances independently. They are presented with the input text and the model outputs. Model responses are listed in random order, with all the model information anonymized. Figure 9 provides a screenshot of the annotation interface. The reported performance in this paper is based on the results from all evaluators.

<sup>7</sup><https://github.com/doccano/doccano>

---

**Input Text:** It is worth mentioning that when the reaction of 2-cyano-N,N-dimethylethanethioamide (1s) with 3-phenylpropionaldehyde (2a) was carried out at room temperature under nitrogen for 5 h, an aldol condensation product 5a was isolated in 73% yield, which can be further converted to 2-aminothienyl ether 3sa in 85% yield under the optimized reaction conditions.

---

**ReactIE (Significant Error)**

Reaction 1:

Product: 5a

Reactants: (1s), (2a), 2-cyano-N,N-dimethylethanethioamide, 3-phenylpropionaldehyde

Reaction type: aldol condensation

Catalyst: nitrogen

Temperature: room

Time: 5 h

Yield: 73%

Reaction 2:

Product: 3sa

Reactants: (1s), (2a), 2-cyano-N,N-dimethylethanethioamide, 3-phenylpropionaldehyde

Yield: 85%

---

**GPT-4 (Minor Issue)**

Reaction 1

Product: aldol condensation product 5a

Reactant: 2-cyano-N,N-dimethylethanethioamide (1s)

Reactant: 3-phenylpropionaldehyde (2a)

Atmosphere: nitrogen

Temperature: room temperature

Time: 5 h

Yield: 73% (aldol condensation product 5a)

Reaction 2

Product: 2-aminothienyl ether 3sa

Reaction type: optimized reaction conditions

Yield: 85% (2-aminothienyl ether 3sa)

---

**REACTION MINER (Perfect)**

Reaction 1

Product: 5a

Reactant: 2-cyano-N,N-dimethylethanethioamide (1s), 3-phenylpropionaldehyde (2a)

Atmosphere: nitrogen

Reaction type: aldol condensation

Temperature: room temperature

Time: 5 h

Yield: 73% (aldol condensation product 5a)

Reaction 2

Product: 2-aminothienyl ether 3sa

Reactant: 5a

Yield: 85% (2-aminothienyl ether 3sa)

---

Table 2: Full text of case study in Figure 5. The blue text indicates the results of the human evaluation, and the red text represents the parts of the model output that are problematic. In addition, the output of GPT-4 misses a few reaction roles, such as the “Reactant” in Reaction 2.

|  |  |  |  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
|--|--|--|--|-----------------------------|----------------------------|--------------------|-----------------------|----------------------|--------------------------|------------------------|------------------|----------------------|----------------|-------------------|------------------------------|-----------------------------|---------------------------------------|--------------------------------|-------------------|-----------------------|--|
| <b>Instruction</b>   | Please help me with a chemistry-related task which is divided into two steps: First, generate a paragraph in a scientific paper, which introduces one or multiple specific chemical reactions. Second, extract the information of all chemical reactions one by one from the generated paragraph. Completing these two steps generates an instance with paragraphs and a corresponding action list. Now please help me generate 5 instances.   |  |  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| <b>Roles Definition</b>  | <p>Specifically, each reaction should include several roles and their corresponding arguments. The roles are predefined attributes involved in the reaction while the arguments are the specific spans extracted from the paragraph that are corresponding to their roles. Here, we list all the reaction roles as below:</p> <table border="0"> <tr> <td>(1) Product: [def]</td> <td>(2) Reactant: [def]</td> <td>(3) Catalyst: [def]</td> <td>(4) Workup reagents: [def]</td> <td>(5) Solvent: [def]</td> </tr> <tr> <td>(6) Atmosphere: [def]</td> <td>(7) Inhibitor: [def]</td> <td>(8) Reaction type: [def]</td> <td>(9) Temperature: [def]</td> <td>(10) Time: [def]</td> </tr> <tr> <td>(11) Pressure: [def]</td> <td>(12) PH: [def]</td> <td>(13) Speed: [def]</td> <td>(14) Vacuum condition: [def]</td> <td>(15) Light condition: [def]</td> </tr> <tr> <td>(16) Cooling/Heating Condition: [def]</td> <td>(17) Spectroscopic data: [def]</td> <td>(18) Yield: [def]</td> <td>(19) Procedure: [def]</td> <td></td> </tr> </table>  | (1) Product: [def]   | (2) Reactant: [def]  | (3) Catalyst: [def]         | (4) Workup reagents: [def] | (5) Solvent: [def] | (6) Atmosphere: [def] | (7) Inhibitor: [def] | (8) Reaction type: [def] | (9) Temperature: [def] | (10) Time: [def] | (11) Pressure: [def] | (12) PH: [def] | (13) Speed: [def] | (14) Vacuum condition: [def] | (15) Light condition: [def] | (16) Cooling/Heating Condition: [def] | (17) Spectroscopic data: [def] | (18) Yield: [def] | (19) Procedure: [def] |  |
| (1) Product: [def]   | (2) Reactant: [def]  | (3) Catalyst: [def]  | (4) Workup reagents: [def]   | (5) Solvent: [def]          |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| (6) Atmosphere: [def]  | (7) Inhibitor: [def]   | (8) Reaction type: [def]   | (9) Temperature: [def]   | (10) Time: [def]            |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| (11) Pressure: [def]   | (12) PH: [def]   | (13) Speed: [def]  | (14) Vacuum condition: [def]   | (15) Light condition: [def] |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| (16) Cooling/Heating Condition: [def]  | (17) Spectroscopic data: [def]   | (18) Yield: [def]  | (19) Procedure: [def]  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| <b>Output Format</b>   | <p>For each instance, the output of our tasks should be in this format:</p> <p>Paragraph:<br/>[the generated text describing chemical reactions]</p> <p>Reaction List:<br/>Reaction 1<br/>[the role-argument pairs of the first reaction. Note that the yield should be the its value followed by the corresponding product.]<br/>...<br/>Reaction n<br/>[the role-argument pairs of the n-th reaction (if any)]</p>   |  |  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| <b>Demonstration ( 3 samples )</b>   | <p>To clearly explain these tasks, we provide the following examples:</p> <p>Instance 1<br/>Paragraph:<br/>Reactions of Zr derivatives such as 8 with [Ph3C][B(C6F5)4] in C6D5Br (at -20 and 20 °C, in absence and presence of d8-THF) were also performed. Although complicated mixtures were similarly produced, the generation of significant amounts of Ph3CCH2Ph were nonetheless observed (CH2 singlet appears at 4.04 ppm in C6D5Br [and 4.03 ppm in C6D5Br containing 3 drops of d8-THF]), suggestive of benzyl abstraction and benzyl cation formation (rather than trityl attack at the C(σ-aryl) atom) like that reported for the Zr-[O,N,C(σ-naphthyl)] analogue.</p> <p>Reaction List:<br/>Reaction 1<br/>Product: Ph3CCH2Ph<br/>Reactant: Zr derivatives such as 8, [Ph3C][B(C6F5)4]<br/>Solvent: C6D5Br containing 3 drops of d8-THF<br/>Reaction type: benzyl abstraction, benzyl cation formation<br/>Temperature: -20 and 20 °C<br/>Yield: significant amounts (Ph3CCH2Ph)</p> <p>Instance 2<br/>Paragraph:<br/>Treatment of CyPBn-Cy with NiCl2(DME) in THF afforded (CyPBn-Cy)NiCl2, which is obtained as the dichloromethane solvate upon workup (68%) on the basis of elemental analysis, NMR spectroscopic, and X-ray crystallographic data. This material in turn was converted into (CyPBn-Cy)Ni(o-tol)Cl upon treatment with (o-tol)MgCl in THF and subsequently isolated as an analytically pure solid (95%).</p> <table border="0"> <tr> <td>Reaction List:<br/>Reaction 1<br/>Product: (CyPBn-Cy)NiCl2<br/>Reactant: CyPBn-Cy, NiCl2(DME)<br/>Solvent: THF<br/>Yield: 68% ((CyPBn-Cy)NiCl2)</td> <td>Reaction 2<br/>Product: (CyPBn-Cy)Ni(o-tol)Cl<br/>Reactant: (CyPBn-Cy)NiCl2, (o-tol)MgCl<br/>Solvent: THF<br/>Yield: 95% ((CyPBn-Cy)Ni(o-tol)Cl)</td> </tr> </table> <p>Instance 3<br/>Paragraph:<br/>1.44 ml (12.5 mmols) of benzoyl chloride, 1.88 ml (12.5 mmols) of N-benzyl dimethylamine and 0.0281 g (0.125 mmol) of palladium acetate are added to 25 ml of toluene in a pressure apparatus constructed of glass. The apparatus is flushed with ethylene in order to remove the air. Ethylene is then injected at 10 bar and the mixture is stirred for 4 hours at 120° C. 55% of styrene and 9% of trans-stilbene are formed.</p> <p>Reaction List:<br/>Reaction 1<br/>Product: styrene, trans-stilbene<br/>Reactant: benzoyl chloride, N-benzyl dimethylamine<br/>Catalyst: palladium acetate<br/>Solvent: toluene<br/>Temperature: 120 °C<br/>Time: 4 hour<br/>Pressure: 10 bar<br/>Yield: 55% (styrene), 9% (trans-stilbene)</p> | Reaction List:<br>Reaction 1<br>Product: (CyPBn-Cy)NiCl2<br>Reactant: CyPBn-Cy, NiCl2(DME)<br>Solvent: THF<br>Yield: 68% ((CyPBn-Cy)NiCl2) | Reaction 2<br>Product: (CyPBn-Cy)Ni(o-tol)Cl<br>Reactant: (CyPBn-Cy)NiCl2, (o-tol)MgCl<br>Solvent: THF<br>Yield: 95% ((CyPBn-Cy)Ni(o-tol)Cl) |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| Reaction List:<br>Reaction 1<br>Product: (CyPBn-Cy)NiCl2<br>Reactant: CyPBn-Cy, NiCl2(DME)<br>Solvent: THF<br>Yield: 68% ((CyPBn-Cy)NiCl2) | Reaction 2<br>Product: (CyPBn-Cy)Ni(o-tol)Cl<br>Reactant: (CyPBn-Cy)NiCl2, (o-tol)MgCl<br>Solvent: THF<br>Yield: 95% ((CyPBn-Cy)Ni(o-tol)Cl)   |  |  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |
| <b>Trigger</b>   | Following the above examples, please help me with this task, i.e. generate 5 instances containing paragraph and reaction list. Remember to strictly follow the output format. Ensure that all arguments of reactions are the real spans extracted from the paragraph and should be faithful to the original text. At least 1 of the 5 generated instances should contain multiple chemical reactions.  |  |  |                             |                            |                    |                       |                      |                          |                        |                  |                      |                |                   |                              |                             |                                       |                                |                   |                       |  |

Figure 8: Prompt used for role enrichment. Specifically in “Role Definitions”, [def] is the placeholder of definitions for all the reaction roles. For a complete list of definitions, please refer to Appendix C.

1 of 400 |< > |

Correct and satisfying   Acceptable but has minor errors or imperfections   Relevant with significant errors   Irrelevant or completely invalid

**\*\*Text\*\***

When the condensation was carried out with malonic acid (2e), 2-benzylidene-malonic acid (3ae) was obtained in 79% yield without a Doebner-type decarboxylation. Also, when the reaction of malononitrile (2f) was carried out in toluene, only 15% (NMR yield) of the product was obtained. It was interesting that the use of N,N-dimethylformamide (DMF) as a solvent instead of toluene successfully improved the chemical yield of alkene 3af to 86%. When the condensation of 1a with the cyclic 1,3-diketone dimedone (2g) was conducted under the optimal conditions, 1:2 adduct 7 (a xanthenedione derivative) was isolated as the sole product, which otherwise would have been produced via a further Michael addition of 2g to the first Knoevenagel adduct and a subsequent intramolecular cyclodehydration.

**\*\*Output\*\***

Reaction 1:  
 Product: 2-benzylidene-malonic acid  
 Reactants: (2e, malonic acid)  
 Reaction type: condensation  
 Yield: 79%

Reaction 2:  
 Product: 3ae  
 Reactants: (2e)  
 Reaction type: condensation

Progress

|          |     |
|----------|-----|
| Total    | 400 |
| Complete | 0   |

0%

| Key               | Value |
|-------------------|-------|
| No data available |       |

Figure 9: Annotation interface for human evaluation. The predictions from different models present in random order and the model information being anonymized. Our expert evaluators are required to read the input text, and then select the rating for the model's outputs from four options for the extracted results.

# CHAMP: Efficient Annotation and Consolidation of Cluster Hierarchies

Arie Cattan<sup>1</sup> Tom Hope<sup>2,3</sup> Doug Downey<sup>2</sup> Roy Bar-Haim<sup>4</sup>  
Lilach Eden<sup>4</sup> Yoav Kantor<sup>4</sup> Ido Dagan<sup>1</sup>

<sup>1</sup>Computer Science Department, Bar Ilan University

<sup>2</sup>Allen Institute for Artificial Intelligence

<sup>3</sup>School of Computer Science, The Hebrew University of Jerusalem

<sup>4</sup>IBM Research

arie.cattan@gmail.com

## Abstract

Various NLP tasks require a complex hierarchical structure over nodes, where each node is a cluster of items. Examples include generating entailment graphs, hierarchical cross-document coreference resolution, annotating event and subevent relations, etc. To enable efficient annotation of such hierarchical structures, we release CHAMP, an open source tool allowing to incrementally construct both clusters and hierarchy simultaneously over any type of texts. This incremental approach significantly reduces annotation time compared to the common pairwise annotation approach and also guarantees maintaining transitivity at the cluster and hierarchy levels. Furthermore, CHAMP includes a consolidation mode, where an adjudicator can easily compare multiple cluster hierarchy annotations and resolve disagreements.

 <https://github.com/ariecattan/champ>

## 1 Introduction

In numerous annotation tasks, the annotator needs to perform individual and independent decisions. Such tasks include Named Entity Recognition (NER), text categorization and part-of-speech tagging, among others (Stenetorp et al., 2012; Yimam et al., 2013; Samih et al., 2016; Yang et al., 2018; Tratz and Phan, 2018; Mayhew and Roth, 2018). However, certain annotation tasks are more demanding because they involve the construction of a complex structure that must satisfy global constraints. One such complex structure is clustering, where annotated clusters must respect the equivalence relation. Specifically, if items A and B belong to the same cluster, and items B and C also belong to the same cluster, then A and C must belong to the same cluster as well. Another prominent example of a global structure is hierarchy, where typically, if A is an ancestor of B and B is an ancestor of C, then A must also be an ancestor of C.

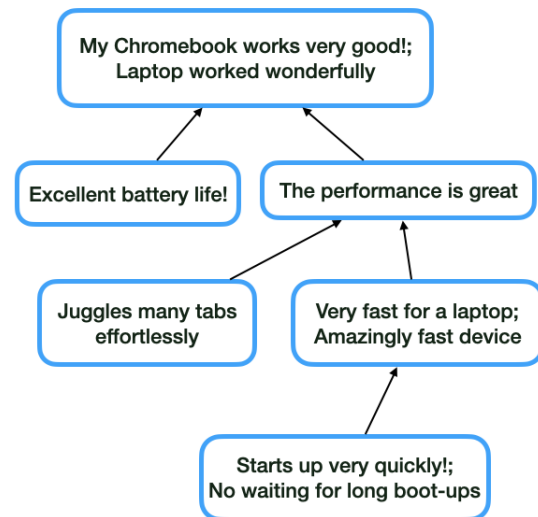


Figure 1: Example of hierarchy of clusters from THINKP (Cattan et al., 2023). Nodes group similar statements together and arrows represent child-parent relations, relating specific statements to more general ones.

In this work, we focus on annotating a *hierarchy of clusters*, a global structure that combines the constraints of both clustering and hierarchy, thereby posing further challenges. In this hierarchy, nodes are clusters of (text) items, where each node can have at most a single parent, as illustrated in Figure 1. Annotating a hierarchy of clusters is relevant for a multitude of tasks, such as hierarchical cross-document coreference resolution (Cattan et al., 2021), structured summarization as a hierarchy of key points (Cattan et al., 2023), entailment graph construction (Berant et al., 2012) and event-subevent relations detection (O’Gorman et al., 2016; Wang et al., 2022). While there are some annotation tools for annotating either clustering or a hierarchy (§2.1), to the best of our knowledge there is no available tool allowing to annotate a hierarchy of clusters simultaneously within the same tool.

To address this need, we introduce CHAMP (Cluster Hierarchy Annotation for Multiple Participants), an intuitive and efficient tool for annotating a hierarchy of clusters in a globally consistent manner, supporting multiple annotators (§3). Specifically, annotators are presented with input text spans one by one and form *incrementally* and *simultaneously* the clusters and their hierarchy (§3.1).

Additionally to the annotation process, we develop an adjudication mode for easily comparing multiple annotated hierarchies of clusters (§3.2). This mode can be used either by an adjudicator, which is typically a more reliable annotator, or by the original annotators during discussions to resolve conflicts. Indeed, adjudication is crucial to ensure quality in general (Roit et al., 2020; Klein et al., 2020), and particularly important for our structure, requiring a more challenging global annotation.

We demonstrate the use of CHAMP in two notably different use-cases, both involving annotating hierarchies of clusters: hierarchical cross-document coreference resolution (Cattan et al., 2021) and key point hierarchy (Cattan et al., 2023). In both settings, CHAMP is significantly more efficient than a pairwise annotation approach, in which the relation between each pair of items is annotated independently. Moreover, our consolidation phase enhances the annotation quality, yielding an improvement of 5-6 F1 points (Cattan et al., 2023).

CHAMP was implemented on top of COREFI (Bornstein et al., 2020), which was initially designed for coreference, and allowed only standard (non-hierarchical) annotation. CHAMP includes a WebComponent, which can easily be embedded into any HTML page, including popular crowdsourcing platforms such as Amazon Mechanical Turk. We also develop an annotation portal (the link appears in our github repository), allowing users to perform online the annotation task and dataset developers to effortlessly compute inter-annotator agreement.

Overall, CHAMP is an intuitive tool for efficiently annotating and adjudicating hierarchies of clusters. We believe that CHAMP will remove barriers when annotating such challenging global tasks and will facilitate future dataset creation.

## 2 Background

### 2.1 Tools for Annotating Global Structures

Certain NLP tasks involve a structure that should be annotated in a global manner due to mutually dependent labels. In this work, we focus on two specific structures: clustering and hierarchy.

A prominent clustering task is coreference resolution, where the goal is to group mention spans into clusters. This implies that if A and B are coreferent and B and C are coreferent, then A and C should also be coreferent. However, early tools for coreference annotation relied on a series of local binary decisions over all possible mention pairs (Stenetorp et al., 2012; Widlöcher and Mahtet, 2012; Landragin et al., 2012; Kopeć, 2014; Chamberlain et al., 2016). In contrast, cluster-based tools aim for global annotation by directly assigning mentions to clusters (Ogren, 2006; Girardi et al., 2014; Reiter, 2018; Oberle, 2018; Aralikatte and Søgaard, 2020; Bornstein et al., 2020; Gupta et al., 2023). Among these cluster-based tools, COREFI (Bornstein et al., 2020) stands out for its beneficial features that enable cost-effective and efficient annotation. These features include quick keyboard operations (instead of slow drag-and-drop), an onboarding mode for training annotators on the task, and a reviewing mode that facilitates systematic review and quality improvement of a given annotation (as described in §2.2).

Some other tasks such as taxonomy induction and entailment graph construction also involve structures (e.g., graphs, DAG, hierarchy) that impose global transitivity constraints. For example, if a taxonomy includes the relationships “A is a kind of B” and “B is a kind of C”, then it follows that A must also be a kind of C. Yet, for example, Berant et al. (2011) annotated an entailment graph dataset by annotating all possible edges between predicates, resulting in a complexity of  $\mathcal{O}(n^2)$ . Subsequent works follow the pairwise approach but apply some heuristics for reducing the number of annotations (Levy et al., 2014; Kotlerman et al., 2015). Closely related to taxonomy, the Redcoat annotation tool (Stewart et al., 2019) allows to annotate hierarchical entity typing, while allowing to modify the hierarchy during annotation.

To the best of our knowledge, there is no available tool that supports joint annotation of a hierarchy of clusters, as proposed in CHAMP.



## 2.2 Consolidation of Multiple Annotations

To promote quality, datasets often rely on multiple annotators per instance, especially when the annotation is obtained via crowdsourcing. Then, the annotations can be combined either *automatically*, using simple majority vote or more sophisticated aggregation techniques (Dawid and Skene, 1979; Raykar et al., 2010; Hovy et al., 2013; Passonneau and Carpenter, 2014; Paun et al., 2018), or *manually*, by asking the annotators themselves or a more reliable annotator to adjudicate and resolve annotation disagreements (Pradhan et al., 2012; Roit et al., 2020; Pyatkin et al., 2020; Klein et al., 2020). However, those aggregation methods were mostly investigated for classification tasks where each instance can be annotated independently, but not for global tasks, like those discussed above (§2.1).

To the best of our knowledge, COREFI (Bornstein et al., 2020) is the only annotation tool that supports *manual* reviewing of a global structure annotation, specifically for coreference annotation. In this interface, the reviewer is shown the annotated mentions one by one along with the original annotator’s cluster assignment. The reviewer can then decide whether to retain the original annotation or to make a different clustering assignment. However, showing the original cluster assignment of each mention in turn is not straightforward, because earlier reviewer decisions may have deviated from the original clustering annotation. For instance, consider a scenario where the original annotator creates a cluster with the mentions  $x, y, z$ . Subsequently, the reviewer decides that  $y$  should not be linked to  $x$  but should instead form a new cluster. At this point, when the reviewer encounters the mention  $z$ , it becomes uncertain whether it should be considered by the original annotation as linked with  $x$  or  $y$ . To address this issue, when the reviewer is shown a mention  $m$ , the candidate clusters implied by the original annotation becomes the *set* of clusters in the current reviewer’s clustering configuration that include at least one of the previously annotated antecedents of  $m$  according to the original annotation.

While the reviewing mode in COREFI is effective, an important limitation is that it enables reviewing only a single annotation, not supporting the consolidation of multiple annotations, as common in NLP annotation setups. We address this need in CHAMP by supporting consolidation of multiple annotations (§3.2).

## 3 CHAMP

We present CHAMP, a new tool for annotating a *hierarchy of clusters*. To annotate such a structure, the annotators are provided with a list of input spans, denoted as  $S = \{s_1, \dots, s_n\}$ , that they need to group into disjoint clusters of semantically equivalent spans  $\mathcal{C} = \{C_1, \dots, C_k\}$ . In addition, annotators need to form a *directed* forest  $G = (\mathcal{C}, E)$ , constituting a Directed Acyclic Graph (DAG) in which every node—representing the cluster  $C_i$ —has no more than one parent. Within this structure, each edge  $e_{ij}$  represents a hierarchical relation between clusters  $C_i \rightarrow C_j$ , signifying that  $C_i$  is a child of  $C_j$ . Considering the example in Figure 1, the cluster *{Starts up very quickly, No waiting for long boot-ups}* is more specific than the cluster *{Very fast for a laptop, Amazingly fast device}*. Importantly, input spans can be standalone spans (as in Figure 1) or appear within a surrounding context. For the remainder of this section, we will focus on demonstrating CHAMP using standalone spans, while an example featuring spans within context is provided in Appendix A.

We next describe the core annotation interface (§3.1), and then present the *adjudication* mode, which allows to effectively compare multiple annotations and build a consolidated hierarchy of clusters (§3.2).

### 3.1 Cluster Hierarchy Annotation

Figure 2 shows the annotation interface in CHAMP.

A naive approach for supporting the annotation of a hierarchy of clusters would involve two separate steps: (1) cluster input spans and (2) construct a hierarchy over the fixed annotated clusters. Although straightforward, this method lacks the flexibility for annotators to modify the clustering annotation while simultaneously working on the hierarchy. This inflexibility is problematic since typically many annotation decisions fall at the intersection of clustering, which reflects semantic equivalences, and hierarchy, which denotes the relationships between more general and specific clusters (e.g., *Takes a long time for check in* vs. *The absolute worst check in process anywhere*). Moreover, employing two separate annotation steps would burden annotators with the additional challenge of remembering the context of each cluster during hierarchy annotation.

Therefore, we propose an *incremental* approach for annotating both the clustering and the cluster hi-

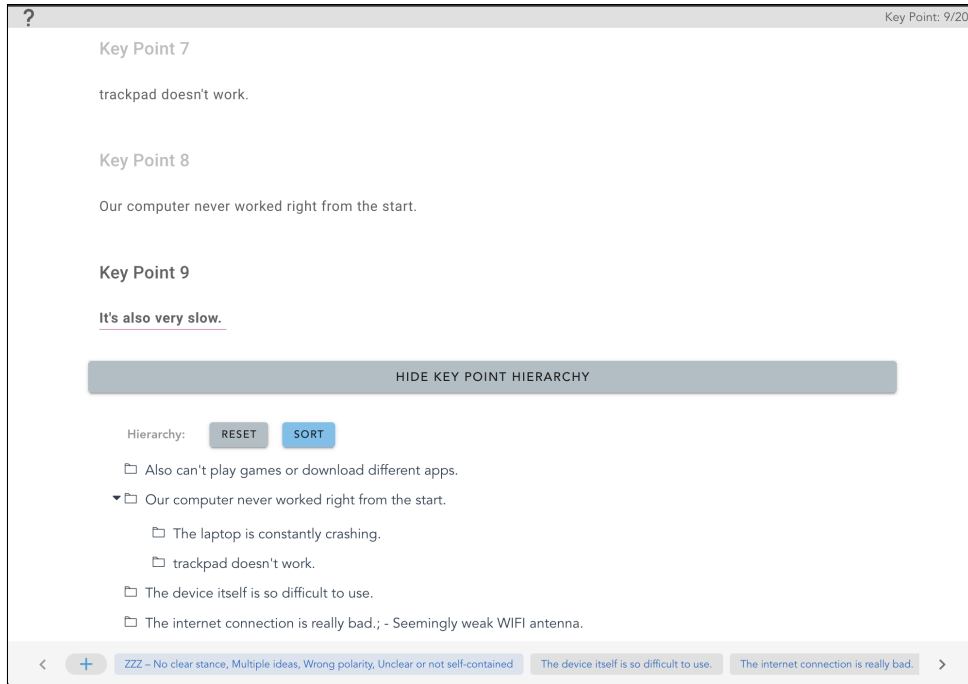


Figure 2: User interface for annotating both clustering and hierarchical relations between clusters. The current statement to assign is underlined in purple: “It’s also very slow”. The annotator can decide whether to add it to an existing cluster, in which case it will be concatenated in the display of the corresponding node in the hierarchy, separated by “;”, or to open a new cluster, in which case a new node will be automatically added to the hierarchy, initiated under the root.

erarchy together as a single annotation task, which we develop upon COREFI (Bornstein et al., 2020). At initialization, the first span is automatically assigned to the first cluster  $C_1$  and to a corresponding node in the hierarchy. Then, for each subsequent span  $s$ , the annotator first decides its cluster assignment, by choosing whether to assign  $s$  to an existing or a new cluster. In the latter case, a new node is automatically created in the hierarchy under the root and the annotator can drag it to its right position in the current hierarchy. Considering the example in Figure 2, the current span to annotate  $s$  is “It’s also very slow” (underlined in purple), the current clusters  $\mathcal{C}$  are shown in the cluster bank (in the footer of the screen), and the current hierarchy is shown in the lower portion of the window.

Importantly, when the annotator re-assigns a previously assigned span to another cluster, CHAMP will automatically update nodes and relations in the hierarchy. Keeping in sync cluster assignments and hierarchy is not trivial because different clustering modifications will have different effects on the resulting hierarchy. In particular, we consider the following cases of re-assigning the span  $s$ :

1. From a *singleton* cluster  $C_i$  to a cluster  $C_j$ :  $s$  will be added to  $C_j$  and  $C_i$ ’s children will move

under  $C_j$ .

2. From a *non-singleton* cluster  $C_i$  to a cluster  $C_j$ :  $s$  will be added to  $C_j$  but  $C_i$ ’s children will stay under  $C_i$ .

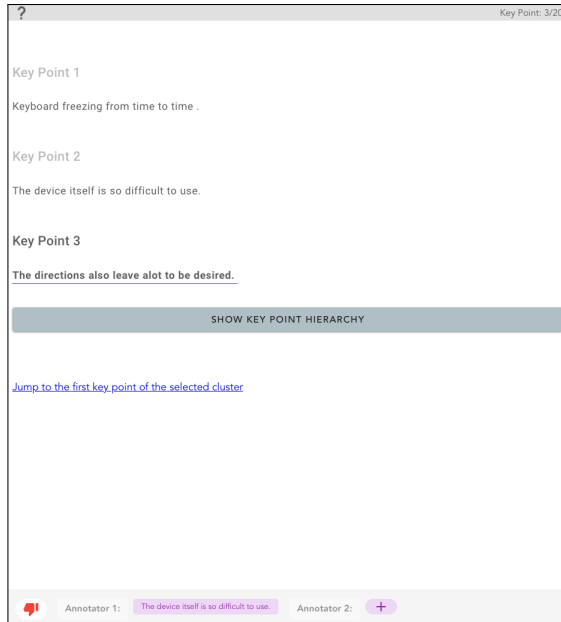
3. From a cluster  $C_i$  to a new singleton cluster: a new node  $C_j$  will be created in the hierarchy and will be initially situated as a sibling of  $C_i$ .<sup>1</sup> Annotators can then drag it to its desired place.

This hierarchy update procedure is a key ingredient for enabling the annotation of hierarchy of clusters as a single task.

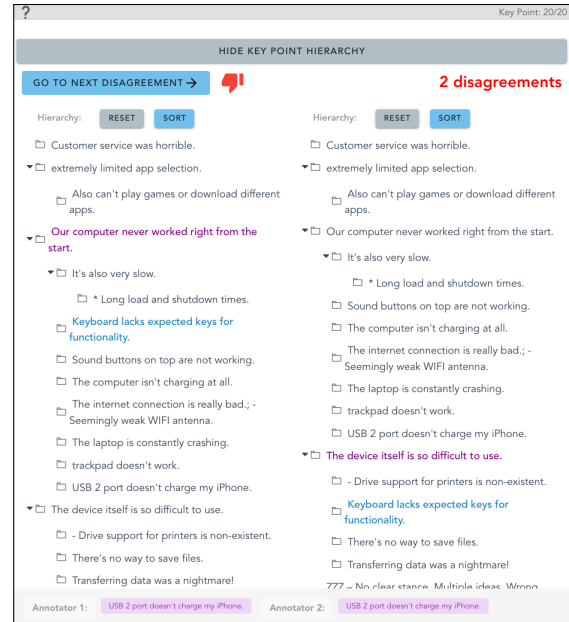
### 3.2 Adjudication

In order to facilitate the manual adjudication of multiple hierarchy annotations by different annotators, we added an adjudication mode within CHAMP that supports easily identification and resolution of disagreements between any number of annotations. This mode can be used by an adjudicator, which is usually a more reliable annotator, or by the original annotators during discussions to resolve conflicts.

<sup>1</sup>We take this approach because, when annotators re-assign  $s$  to a standalone cluster, their intention is not to eliminate the hierarchical relationship between  $s$  and its parent cluster.



(a) Clustering consolidation. The thumb-down at the bottom left of the screen indicates a clustering disagreement between the annotators for the span “*The directions also leave a lot to be desired*”. Annotator A1 assigned it to “*The device itself is so difficult to use*” while annotator A2 created a new cluster, as indicated in purple.



(b) Hierarchy consolidation. The red thumb-down near the “Go to next disagreement” button indicates a hierarchy disagreement for the node “*Keyboard lacks expected keys for functionality*”. Annotator A1 placed it under “*Our computer never worked right from the start*”, while A2 placed it under “*The device itself is so difficult to use.*”

Figure 3: Adjudication of multiple annotations of hierarchy of clusters.

Comparing multiple annotations of a hierarchy of clusters can be challenging due to variations in annotators’ clustering assignments, leading to different sets of nodes in the respective hierarchies. To illustrate this issue, consider a scenario where annotator A1 annotates the relation  $\{s_1, s_2, s_4\} \rightarrow \{s_3, s_5\}$ , while A2 annotates  $\{s_1, s_2, s_6\} \rightarrow \{s_3\}$  and  $\{s_4\} \rightarrow \{s_5\}$ . The two hierarchies have similarities (e.g. both cluster  $s_1$  and  $s_2$  together and have  $s_5$  as a parent of  $s_4$ ) but differ in other ways, making their adjudication process non-trivial.

To tackle this problem, we decoupled the adjudication process into two consecutive stages, adjudicating separately clustering and hierarchy decisions, as illustrated in Figure 3.

In the first step, the adjudicator is shown the annotated spans in a sequential manner, along with the cluster assignments of each of the original annotations. To achieve this, we leverage the reviewing procedure that COREFI applies for reviewing a single clustering annotation (§2.2), implement it separately to each original annotation. We then present to the adjudicator a set of candidate clusters per original annotation. These sets of candidates are displayed in purple at the bottom of the screen, as illustrated in Figure 3a.

It should be pointed out here that resolving a cluster assignment disagreement means that the adjudicator alters the assignment for at least one of the annotators. Therefore, we apply the hierarchy update procedure (§3.1) to the modified annotations, in order to update accordingly the involved cluster nodes and their hierarchical relations. Considering the example in Figure 3a with a clustering disagreement for the span “*The directions also leave a lot to be desired*” ( $s_1$ ). In this instance, annotator A1 has merged it with “*The device itself is so difficult to use*” ( $s_2$ ), while annotator A2 has designated it as a singleton cluster in the hierarchy, as highlighted by the purple ‘+’ button. If the adjudicator follows A1’s decision, A2’s hierarchy will be restructured to combine spans  $\{s_1, s_2\}$  into the same cluster. Conversely, siding with A2’s decision will separate  $s_2$  from  $s_1$  in A1’s hierarchy. This automatic process ensures that the modified hierarchies will include the exact same set of nodes (clusters)  $\mathcal{C}$  at the end of the clustering consolidation step.

In the second step of hierarchy adjudication, as the sets of nodes  $\mathcal{C}$  in the hierarchies of all annotators are identical, a disagreement arises when a node  $C_i \in \mathcal{C}$  has a different direct parent in different hierarchies. To efficiently identify such dis-

crepancies, the adjudicator can click on the “Go To Next Disagreement” button, which highlights the node  $C_i$  in blue along with its direct parent in violet on all input hierarchies. As shown in Figure 3b, for instance, the node “*Keyboard lacks expected keys for functionality*” was placed under “*Our computer never worked right from the start*” by A1, and under “*The device itself is so difficult to use*” by A2. The adjudicator then decides the correct hierarchical relation, manually updates the other hierarchies accordingly, and moves on to the next disagreement. Once all hierarchical disagreements have been resolved, the adjudicator can confidently submit the obtained consolidated hierarchy.

## 4 Applications

We used CHAMP for annotating datasets for two different tasks that require annotating of hierarchy of clusters:

1. SciCo (Cattan et al., 2021), a dataset for the task of hierarchical cross-document coreference resolution (H-CDCR). In this dataset, the inputs are paragraphs from computer science papers with highlighted mentions of scientific concepts, specifically mentions of tasks and methods. The goal is to first cluster all mentions that refer to the same concept (e.g., *categorical image generation*  $\leftrightarrow$  *class-conditional image synthesis*) and then infer the referential hierarchy between the clusters (e.g., *categorical image generation*  $\rightarrow$  *image synthesis*).
2. THINKP (Cattan et al., 2023), a recent benchmark of key point hierarchies, where each key point is a concise statement relating to a particular topic (Bar-Haim et al., 2020). Key point hierarchies were proposed as a novel structured representation for large scale opinion summarization. The nodes in these graphs group statements conveying the same opinion (e.g., *the cleaning crew is great!*  $\leftrightarrow$  *housekeeping is fantastic*) while the edges indicate hierarchical specification-generalization relationships between nodes (e.g., *housekeeping is fantastic*  $\rightarrow$  *the personnel is great*). The entailment graphs in THINKP are designed in a hierarchical form, where each node has at most a single parent.

Despite the different nature of these tasks and their unit of annotation (i.e., standalone statements vs. concept spans in context), we seamlessly leveraged CHAMP for both with minimal effort (using a simple JSON configuration schema), as both tasks involve annotating a hierarchy of clusters.

In our experiments, we observed that annotating or consolidating a hierarchy of clusters for fifty statements takes approximately one hour (Cattan et al., 2023). In contrast, collecting annotations for all possible pairs, as commonly done in prior datasets for entailment graphs (Berant et al., 2011), would have been much more expensive since it would require at least 1225 decisions on average for our data, which would obviously take much more than one hour. Furthermore, unlike the pairwise annotation approach, our incremental method for constructing a hierarchy of clusters guarantees that the resulting annotation will respect the global constraint of transitivity. Finally, our experiments also revealed that the consolidation mode significantly enhances human performance, yielding a gain of 5-6 F1 points (Cattan et al., 2023).

## 5 Implementation Details and Release

We implement CHAMP on top of COREFI (Bornstein et al., 2020), using the `Vue.js` framework, that we open source under the permissive MIT License. Following COREFI, we release CHAMP as a WebComponent, which can easily be embedded into any HTML page, including popular crowdsourcing platforms such as Amazon Mechanical Turk. Both the annotation and consolidation processes share the same interface and are easily configurable using a straightforward JSON schema. We also develop an annotation portal where users can upload a configuration file (either for annotation or adjudication), perform the annotation task and download it upon completion. This portal also provides the capability to upload multiple annotation files from various annotators and to compute the inter-annotator agreement. As such, CHAMP is not only easy-to-use for annotators, but it is also easy to setup and manage for dataset developers.

## 6 Conclusion

This paper aims to foster research on global annotation tasks by introducing CHAMP, an efficient tool designed for annotating a *hierarchy of clusters*. This annotation tool also incorporates an adjudication mode that conveniently supports identification and consolidation of annotators’ disagreements. As CHAMP enables efficient and high-quality annotation, we believe that it will facilitate the creation of datasets for various tasks involving this complex structure, and will inspire tool development for other global annotation tasks.

## Acknowledgements

This work was supported by the Israel Science Foundation (grant no. 2827/21). Arie Cattan is partially supported by the PBC fellowship for outstanding PhD candidates in data science.

## References

- Rahul Aralikkatte and Anders Søgaard. 2020. [Model-based annotation of coreference](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 74–79, Marseille, France. European Language Resources Association.
- Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. [From arguments to key points: Towards automatic argument summarization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.
- Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. 2012. [Efficient tree-based approximation for entailment graph learning](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 117–125, Jeju Island, Korea. Association for Computational Linguistics.
- Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. [Global learning of typed entailment rules](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 610–619, Portland, Oregon, USA. Association for Computational Linguistics.
- Ari Bornstein, Arie Cattan, and Ido Dagan. 2020. [CoRefi: A crowd sourcing suite for coreference annotation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 205–215, Online. Association for Computational Linguistics.
- Arie Cattan, Lilach Eden, Yoav Kantor, and Roy Bar-Haim. 2023. [From key points to key point hierarchy: Structured and expressive opinion summarization](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 912–928, Toronto, Canada. Association for Computational Linguistics.
- Arie Cattan, Sophie Johnson, Daniel S Weld, Ido Dagan, Iz Beltagy, Doug Downey, and Tom Hope. 2021. [Scico: Hierarchical cross-document coreference for scientific concepts](#). In *3rd Conference on Automated Knowledge Base Construction*.
- Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2016. [Phrase detectives corpus 1.0 crowd-sourced anaphoric coreference](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 2039–2046, Portorož, Slovenia. European Language Resources Association (ELRA).
- A. Philip Dawid and Allan Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of The Royal Statistical Society Series C-applied Statistics*, 28:20–28.
- Christian Girardi, Manuela Speranza, Rachele Sprugnoli, and Sara Tonelli. 2014. [CROMER: a tool for cross-document event and entity coreference](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 3204–3208, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Ankita Gupta, Marzena Karpinska, Wenlong Zhao, Kalpesh Krishna, Jack Merullo, Luke Yeh, Mohit Iyyer, and Brendan O’Connor. 2023. [ezCoref: Towards unifying annotation guidelines for coreference resolution](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 312–330, Dubrovnik, Croatia. Association for Computational Linguistics.
- Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. [Learning whom to trust with MACE](#). In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.
- Ayal Klein, Jonathan Mamou, Valentina Pyatkin, Daniela Stepanov, Hangfeng He, Dan Roth, Luke Zettlemoyer, and Ido Dagan. 2020. [QANom: Question-answer driven SRL for nominalizations](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3069–3083, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Mateusz Kopeć. 2014. [MMAX2 for coreference annotation](#). In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 93–96, Gothenburg, Sweden. Association for Computational Linguistics.
- Lili Kotlerman, Ido Dagan, Bernardo Magnini, and Luisa Bentivogli. 2015. Textual entailment graphs. *Natural Language Engineering*, 21:699 – 724.
- Frédéric Landragin, Thierry Poibeau, and Bernard Victorri. 2012. [ANALEC: a new tool for the dynamic annotation of textual data](#). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 357–362, Istanbul, Turkey. European Language Resources Association (ELRA).
- Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. [Focused entailment graphs for open IE propositions](#).

- In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan. Association for Computational Linguistics.
- Stephen Mayhew and Dan Roth. 2018. **TALen: Tool for annotation of low-resource ENtities**. In *Proceedings of ACL 2018, System Demonstrations*, pages 80–86, Melbourne, Australia. Association for Computational Linguistics.
- Bruno Oberle. 2018. **SACR: A drag-and-drop based tool for coreference annotation**. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Tim O’Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. **Richer event description: Integrating event coreference with temporal, causal and bridging annotation**. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56, Austin, Texas. Association for Computational Linguistics.
- Philip V. Ogren. 2006. **Knowtator: A protégé plug-in for annotated corpus construction**. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Demonstrations*, pages 273–275, New York City, USA. Association for Computational Linguistics.
- Rebecca J. Passonneau and Bob Carpenter. 2014. **The benefits of a model of annotation**. *Transactions of the Association for Computational Linguistics*, 2:311–326.
- Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. **Comparing Bayesian models of annotation**. *Transactions of the Association for Computational Linguistics*, 6:571–585.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. **CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes**. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.
- Valentina Pyatkin, Ayal Klein, Reut Tsarfaty, and Ido Dagan. 2020. **QADiscourse - Discourse Relations as QA Pairs: Representation, Crowdsourcing and Baselines**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2804–2819, Online. Association for Computational Linguistics.
- Vikas Chandrakant Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322.
- Nils Reiter. 2018. **CorefAnnotator - A New Annotation Tool for Entity References**. In *Abstracts of EADH: Data in the Digital Humanities*.
- Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. **Controlled crowdsourcing for high-quality QA-SRL annotation**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.
- Younes Samih, Wolfgang Maier, and Laura Kallmeyer. 2016. **SAWT: Sequence annotation web tool**. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 65–70, Austin, Texas. 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.
- Michael Stewart, Wei Liu, and Rachel Cardell-Oliver. 2019. **Redcoat: A collaborative annotation tool for hierarchical entity typing**. 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 193–198, Hong Kong, China. Association for Computational Linguistics.
- Stephen Tratz and Nhien Phan. 2018. **A web-based system for crowd-in-the-loop dependency treebanking**. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Xiaozhi Wang, Yulin Chen, Ning Ding, Hao Peng, Zimu Wang, Yankai Lin, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. **MAVEN-ERE: A unified large-scale dataset for event coreference, temporal, causal, and subevent relation extraction**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 926–941, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Antoine Widlöcher and Yann Mathet. 2012. **The glozz platform: A corpus annotation and mining tool**. In *Proceedings of the 2012 ACM Symposium on Document Engineering, DocEng ’12*, page 171–180, New York, NY, USA. Association for Computing Machinery.
- Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. **YEDDA: A lightweight collaborative text span annotation tool**. In *Proceedings of ACL 2018, System*

*Demonstrations*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

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, Sofia, Bulgaria. Association for Computational Linguistics.

## **A Appendix**

Figure 4 shows the interface of CHAMP for annotating a hierarchy of clusters over text spans appearing in their context. This example was taken from SCICo.

Mention: 5/18 Document: 3/10

**Document 2**  
[Effective LSTMs for Target-Dependent Sentiment Classification](#) (COLING 2016)

document : Effective LSTMs for Target - Dependent Sentiment Classification  
 Target - dependent sentiment classification remains a challenge : modeling the semantic relatedness of a target with its context words in a sentence . Different context words have different influences on determining the sentiment polarity of a sentence towards the target .

License details : **Sentiment analysis** , also known as opinion mining , is a fundamental task in natural language processing and computational linguistics . Sentiment analysis is crucial to understanding user generated text in social networks or product reviews , and has drawn a lot of attentions from both industry and academic communities .

**Document 3**  
[Domain-Adversarial Training of Neural Networks](#) (J. Mach. Learn. Res. 2016)

One important example is training an image classifier on synthetic or semi - synthetic images , which may come in abundance and be fully labeled , but which inevitably have a distribution that is different from real images [ reference ][ reference ] [ reference ] [ reference ] . Another example is in the context of sentiment analysis in **written reviews** , where one might have labeled data for reviews of one type of product ( e.g. , movies ) , while having the need to classify reviews of other products ( e.g. , books ) . Learning a discriminative classifier or other predictor in the presence of a shift between training and test distributions is known as domain adaptation ( DA ) .

We also observed better accuracies when we initialized the learning of the reverse classifier  $\eta_r$  with the configuration learned by the network  $\eta_l$  . section : Experiments on **Sentiment Analysis Data Sets** We now compare the performance of our proposed DANN algorithm to a standard neural network with one hidden layer ( NN ) described by Equation ( 5 ) , and a Support Vector Machine ( SVM ) with a linear kernel .

HIDE CONCEPT HIERARCHY

Hierarchy:

- ▼  Sentiment analysis
  - aspect - based sentiment analysis

+ aspect - based sentiment analysis Sentiment analysis

Figure 4: User interface for annotating hierarchy of clusters over textual spans that appear within surrounding context.



# PROMPT2MODEL: Generating Deployable Models from Natural Language Instructions

Vijay Viswanathan<sup>\*1</sup>, Chenyang Zhao<sup>\*2†</sup>,  
Amanda Bertsch<sup>1</sup>, Tongshuang Wu<sup>1</sup>, Graham Neubig<sup>1</sup>

<sup>1</sup>Carnegie Mellon University,

<sup>2</sup>BNRIST / Department of Computer Science and Technology, Tsinghua University

## Abstract

Large language models (LLMs) enable system builders today to create competent NLP systems through prompting, where they only need to describe the task in natural language and provide a few examples. However, in other ways, LLMs are a step backward from traditional special-purpose NLP models; they require extensive computational resources for deployment and can be gated behind APIs. In this paper, we propose Prompt2Model, a general-purpose method that takes a natural language task description like the prompts provided to LLMs, and uses it to train a special-purpose model that is conducive to deployment. This is done through a multi-step process of retrieval of existing datasets and pre-trained models, dataset generation using LLMs, and supervised fine-tuning on these retrieved and generated datasets. Over three tasks, we demonstrate that given the same few-shot prompt as input, Prompt2Model trains models that outperform the results of a strong LLM, gpt-3.5-turbo, by an average of 20% while being up to 700 times smaller. We also show that this data can be used to obtain reliable *performance estimates* of model performance, enabling model developers to assess model reliability before deployment. Prompt2Model is available open-source at <https://github.com/neulab/prompt2model>.<sup>1</sup>

## 1 Introduction

Traditionally, building an NLP model from scratch has been a substantial undertaking. An NLP practitioner seeking to solve a new problem would need to define their task scope, find or create data that specifies the intended system behavior, choose a suitable model architecture, train the model, assess its performance through evaluation, and then deploy it for real-world usage (Paley et al., 2022).

<sup>\*</sup>equal contribution.

<sup>†</sup>Work done during an internship at Carnegie Mellon.

<sup>1</sup>Our demo video is posted at [youtu.be/LYYQ\\_EhGd-Q](https://youtu.be/LYYQ_EhGd-Q).

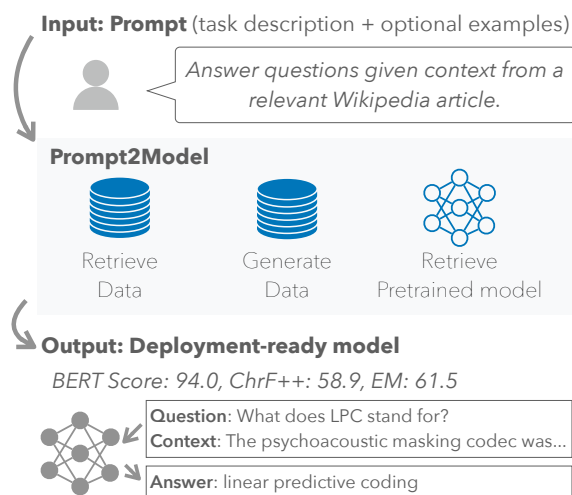


Figure 1: Prompt2Model is a framework for generating a small yet accurate model from a prompt.

LLMs like GPT-3 (Brown et al., 2020; Liu et al., 2023b) offer a lighter-weight paradigm for NLP system construction through “prompting” (Reynolds and McDonell, 2021). Practitioners can now write a prompt specifying the intended system behavior (optionally with a few demonstrations), and ask an LLM to generate a desired output via text completion. This makes it possible to prototype NLP systems rapidly for a variety of applications without writing a single line of code (Floridi and Chiriatti, 2020).

However, there is still a gap between proof-of-concept prototyping — showing LLMs can be prompted for a particular task — and practical deployment. Prompting LLMs can be expensive as they require either a significant amount of computing or access to commercial APIs, and their reliance on the input prompt quality makes them unstable compared to trained models (Min et al., 2022; Bubeck et al., 2023). Because practitioners usually lack annotated data, it is also more challenging for them to evaluate or debug their systems before deployment (Jiang et al., 2022). Additionally, practitioners have expressed concerns about

the high serving cost and slow prediction time associated with using LLMs (Park et al., 2022), and those working in high-stakes domains cannot rely on commercial LLM APIs due to privacy concerns.

In this work, we present Prompt2Model, a system that retains the ability to specify system behavior in a light-weight way through *prompting*, while still resulting in a *deployable special-purpose model*, maintaining all the advantages thereof. Prompt2Model is designed as an automated pipeline that extracts essential task information from users’ prompts and then automatically collects and synthesizes task-specific knowledge through three channels:

- *Dataset retrieval*: Whenever possible, we retrieve task-relevant annotated data (Färber and Leisinger, 2021; Viswanathan et al., 2023).
- *Dataset generation*: We distill knowledge from an LLM (“teacher model”) by employing it to generate a pseudo-labeled dataset (Wang et al., 2021; He et al., 2023; Gudibande et al., 2023).
- *Model retrieval*: Based on the prompt, we identify a pretrained language model appropriate for the user’s intent. This model is fine-tuned and evaluated using the generated and retrieved data.

Prompt2Model is designed to support different instantiations of each of these components. We provide a reference implementation using a gpt-3.5-turbo-based dataset generator, a dataset retriever based on DataFinder (Viswanathan et al., 2023), and a BM25-based model retriever. We evaluate three tasks covering both traditional NLP benchmarks and novel applications and find that, empirically, Prompt2Model sometimes produces small models that outperform gpt-3.5-turbo using the same prompt. On 2 of these 3 tasks, we observe >20 point improvements over gpt-3.5-turbo, despite our finetuned model being up to 700 times smaller. We also find that Prompt2Model can generate effective evaluation datasets. Prompt2Model can serve the following purposes for the community:

1. **A tool for quickly building small and competent NLP systems**: Prompt2Model can produce task-specific models that outperform LLMs in a few hours without any manual data annotation or architecture design. The method bridges the gap between proof-of-concept LLM prototyping and practical deployment.
2. **A testbed for end-to-end, prompt-based model training**: Given Prompt2Model’s ex-

tensible design, it can offer a platform for exploring new techniques in model distillation, dataset generation, synthetic evaluation, dataset retrieval, and model retrieval. Our platform allows studying these components using extrinsic downstream metrics, enabling empirical progress in these research areas.

## 2 Prompt2Model Framework

Prompt2Model provides a framework to automate machine learning pipelines: data collection, model training, evaluation, and deployment. We illustrate our automated pipeline in Figure 2. At the core is our automatic data collection system, leveraging dataset retrieval and LLM-based dataset generation to obtain labeled data. We then retrieve pretrained models and finetune them on the training datasets. Finally, we evaluate trained models on the test datasets and create a web UI for interaction.

Our general-purpose method is designed to be modular and extensible; each component can be implemented differently or disabled by a practitioner. We give an overview of our framework, then in section 3 we describe our reference implementation.

**Prompt Parser** As the primary input to our system, users provide prompts similar to those used for LLMs. These prompts comprise an instruction and, optionally, a few demonstrations of the anticipated behavior. While this open-ended interface is convenient for users, end-to-end ML pipelines may benefit from a *Prompt Parser* that processes this input, such as segmenting the prompt into an instruction and individual demonstrations.

**Dataset Retriever** Given a prompt, we first try to discover existing manually annotated data that support users’ task descriptions. There are several design decisions for the *Dataset Retriever*:

1. What datasets to search against?
2. How to index datasets for search?
3. Which dataset columns are needed for the user’s task, and which columns should be ignored?

We use *DataFinder* (Viswanathan et al., 2023) in our implementation, described in §3.2.

**Dataset Generator** Not all conceivable tasks have any existing relevant annotated data. To support a wider range of tasks, the *Dataset Generator* synthesizes training data from user-specific requirements parsed by the *Prompt Parser*. This presents challenges related to cost efficiency, speed, diver-

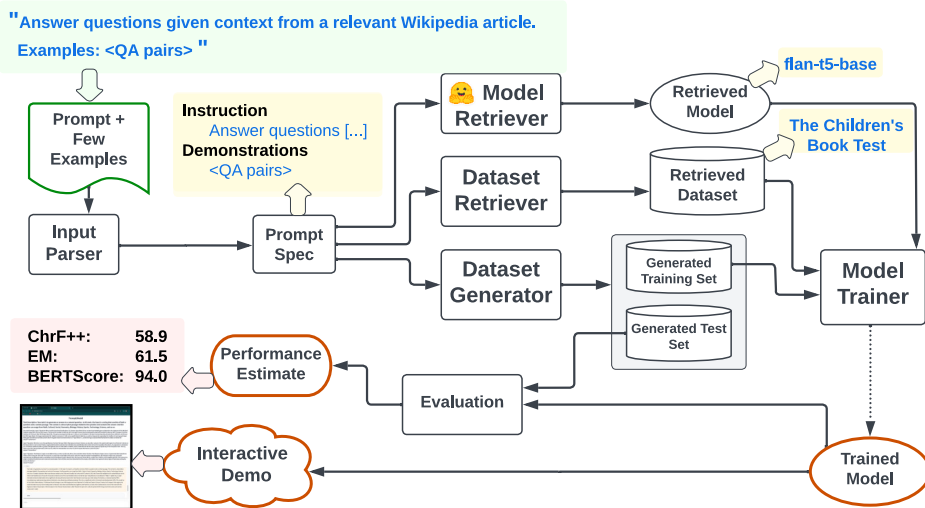


Figure 2: The Prompt2Model architecture seeks to automate the core machine learning development pipeline.

sity, and quality control. Our suggested solution to these challenges is described in §3.3.

**Model Retriever** Besides training data, we must identify an appropriate model to finetune. We cast this as a retrieval problem against model descriptions and metadata such as popularity or tasks supported. The reference implementation of our *Model Retriever*, described in §3.4, searches against models on Hugging Face (Wolf et al., 2020).

**Training** Given retrieved and generated datasets and a pretrained model, we use a *Model Trainer* to finetune the model on a subset of the data.

**Evaluation** After training models on a portion of the retrieved and generated datasets, we give the remaining data to an *Model Evaluator* module. Selecting the correct metrics for an arbitrary task is a difficult problem. We describe our suggested strategies for task-agnostic evaluation in §3.6.

## 2.1 Web App Creation

We include a component called the *Demo Creator* to create a user interface to interact with the model. We briefly describe our implementation in §3.7.

## 3 Reference Implementation

Prompt2Model is designed modularly to support customization of each component in our framework (described in §2), but we have provided a reference implementation to enable immediate adoption.

### 3.1 Prompt Parser

We parse the prompt into instruction and demonstrations fields (shown in Figure 2),

where the instruction represents the task specification and the demonstrations exemplify the desired behavior. We utilize an LLM (OpenAI’s gpt-3.5-turbo-0613 in our experiments) to segment user prompts.

### 3.2 Dataset Retriever

To retrieve datasets for a prompt, we adapt the *DataFinder* system introduced by Viswanathan et al. (2023). We adapted *DataFinder* to user-generated dataset descriptions from Hugging Face Datasets (Lhoest et al., 2021). Once a relevant dataset is identified, the next step is to determine which columns of the dataset correspond to the input and the output specified by the user. As automatically inducing the correct schema for any dataset can be challenging, we adopt a human-in-the-loop approach. We present the top- $k$  datasets, to the user and allow them to either select the most relevant dataset or to state that none are a good fit for their task. We then ask the user to identify the appropriate input and output columns.

### 3.3 Dataset Generator

We engineered our dataset generator to enable speed-optimized generation at a low cost while creating diverse and high-quality examples. Our strategy comprises the following components:

**High-Diversity Few-Shot Prompting** We use automated prompt engineering to generate diverse samples. We expand the user-provided demonstrations with a sample of previously generated examples to promote diversity and avoid duplicates.

**Temperature Annealing** We adjust the sampling temperature from low (favoring determinism) to high (encouraging exploration) proportional to the number of examples already generated to maintain output quality while gradually enabling diversity.

**Minimum Bayes-Risk Decoding** Given that LLM may generate multiple outputs for the same inputs, we use *self-consistency*, a form of Minimum Bayes Risk Decoding (Wang et al., 2022; Bertsch et al., 2023) to select pseudo-labels. Specifically, we create a consensus output for each unique input by selecting the most frequent answer; in the case of ties, we heuristically select the shortest answer.

### 3.4 Model Retriever

To select an appropriate model to fine-tune, we presently limit ourselves to encoder-decoder models on Hugging Face (Wolf et al., 2020) following work showing that encoder-decoder models are more data-efficient for model distillation (Calderon et al., 2023). This restriction still leaves a large set of pretrained models to choose from. Using the user’s instruction as a query, we search against textual descriptions of models on Hugging Face.

This search task is challenging because Hugging Face model descriptions are sparse and contain lots of templatic text, often with only a few words that signify the content of the model. To address this, we follow the HyDE framework (Gao et al., 2023) and use gpt-3.5-turbo to create a *hypothetical model description* given the user’s instructions (with an example shown in Figure 3). Using this as an expanded query, we use BM25 to compute query-model similarity scores (Robertson et al., 1995). For practical purposes, we filter out models whose size exceeds a user-specified threshold (set to 3GB by default). Using the intuition that highly-downloaded models are likely to be high in quality, we finally rank models by:

$$BM25(\text{query}, \text{model}) \cdot \log(\# \text{ of Downloads} + 1).$$

### 3.5 Training

**Dataset Processing** We train the model with up to two datasets- one generated and one retrieved. We treat all datasets as “text-to-text” (Raffel et al., 2020) by textualizing the input columns of each and prepending the user’s instructions to the input.

**Finetuning** We concatenate the retrieved and generated datasets and shuffle them before finetuning. We use the same default hyperparameters

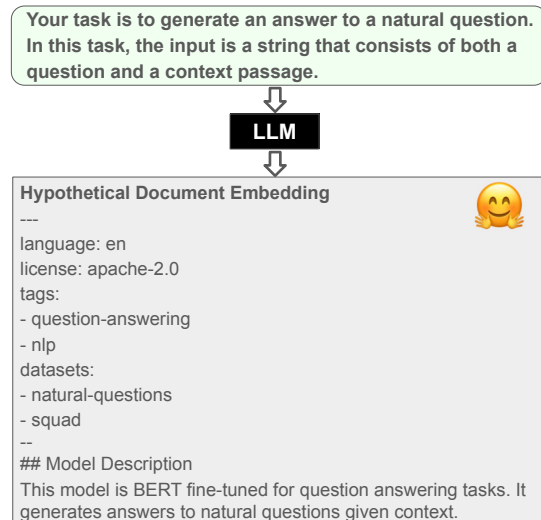


Figure 3: We expand queries to our model retriever by constructing a *hypothetical model description*.

for all tasks.<sup>2</sup> We train with the AdamW optimizer with  $1r = 5e-5$  for 3 epochs, which takes roughly one hour for all tasks.

### 3.6 Evaluation

We automatically evaluate models using three general-purpose metrics: Exact Match, ChrF++ (Popović, 2015), and BERTScore (Zhang et al., 2019). ChrF++ measures character and word overlap. BERTScore captures semantic similarity using embeddings of model outputs and references — we use XLM-R (Conneau et al., 2020) to compute embeddings to support multilingual evaluation.

### 3.7 Web App Creation

We automatically create a graphical user interface to interact with the trained model. This web application, built using Gradio (Abid et al., 2019), can be easily deployed publicly.

## 4 Experimental Setup

**Tasks** As a proof-of-concept, we test our system’s ability to learn a model for three tasks:

- *Machine Reading Question Answering*: We use SQuAD (Rajpurkar et al., 2016) as ground truth to evaluate the setting where pretrained models and training datasets are plentiful.
- *Japanese NL-to-Code*: Code generation from Japanese-language queries is a challenging scenario where prior work exists but no annotated

<sup>2</sup>We empirically find that these default hyperparameters are effective, but we plan on implementing hyperparameter selection using generated validation data in the future.

| Method         | SQuAD<br>(EM) | MCoNaLa<br>(ChrF++) | Temporal<br>(ChrF++) |
|----------------|---------------|---------------------|----------------------|
| Prompt2Model   | 61.5          | 13.1                | 55.2                 |
| w/o Model Ret. | 61.5          | 15.8                | 55.2                 |
| w/o Data Ret.  | 50.2          | 16.6                | N/A                  |
| gpt-3.5-turbo  | 42.1          | 37.3                | 30.7                 |

Table 1: We evaluate the model produced by Prompt2Model on real benchmarks for each test set, compared to gpt-3.5-turbo, also used in our dataset generator. We also examine the effect of removing specific components — model retrieval and dataset retrieval. There are no relevant datasets available for the Temporal task, so we did not use retrieved data there.

data or pretrained models are available. We use MCoNaLa (Wang et al., 2023) for evaluation.

- *Temporal Expression Normalization*: We finally consider a task where no relevant models or data are available. We use the Temporal dataset of Wu et al. (2023) as ground truth for evaluation. Though Prompt2Model offers automated model evaluation (on generated and retrieved datasets), we use real benchmark datasets here to measure Prompt2Model’s ability to train accurate models.

**LLM Baseline** A primary goal of our work is to train small models that can match or outperform LLMs. To measure success towards this goal, we report the performance of gpt-3.5-turbo on each benchmark. We provide gpt-3.5-turbo<sup>3</sup> the same instruction and demonstrations provided to Prompt2Model, for fair comparison.

## 5 Experiment Results

### 5.1 Downstream performance

How effective is Prompt2Model at producing a high-quality model? In Table 1, we evaluated models produced by Prompt2Model and baseline LLM gpt-3.5-turbo, on real benchmark datasets for each task — SQuAD, MCoNaLa, and Temporal. We further examine the effect of removing 2 specific elements of the Prompt2Model pipeline — model retrieval and dataset retrieval.

On 2 of 3 datasets, we find that Prompt2Model produces models that are considerably more accurate than gpt-3.5-turbo. This is remarkable because the retrieved model for SQuAD and Temporal is Flan-T5, which has only 250M parameters.

We observe that Prompt2Model’s performance on MCoNaLa’s Japanese-to-Python task is signif-

<sup>3</sup>We used gpt-3.5-turbo-0613, accessed between July 26 and August 6, 2023.

icantly worse than gpt-3.5-turbo. One explanation is the relatively low diversity of the generated Japanese queries; 45 of 5000 examples are different ways of saying “find the maximum value in a list of numbers“, suggesting that gpt-3.5-turbo may struggle to generate diverse text for non-English languages. Another reason is the lack of an appropriate student model — the retrieved models were trained on either multiple languages or codes, but not both.

### 5.2 Combining retrieved and generated datasets is powerful

Using SQuAD as a case study, we can compare Prompt2Model with a model trained on the same amount of data from the true dataset.<sup>4</sup> Our prompt for Prompt2Model is a description of the SQuAD passage-level question-answering task (Figure 1), and we exclude SQuAD from the retrieved datasets. We evaluate models finetuned on:

1. 3k examples from the closest retrieved dataset<sup>5</sup>
2. 3k examples generated by Prompt2Model
3. 1. + 2. (i.e. the full Prompt2Model pipeline)
4. 3k examples from SQuAD (analogous to the user custom-annotating data for a task).

Table 2 shows the results across these settings. While using only retrieved or generated data causes a reduction in performance, combining these two methods provides similar performance to using a subset of the original SQuAD. Compared to custom-annotating a subset of SQuAD for this task, Prompt2Model allows for *similar performance at less than 1% of the cost*.

### 5.3 Our generated evaluation data can identify real modeling improvements

High-quality generated data should *discriminate* between multiple candidate models to select a model that will perform well downstream. We finetune various models on a generated dataset and rank their performance on generated test data and the test data from the target (real) dataset. We evaluate Kendall’s rank correlation (Kendall, 1938) between the two rankings to determine if our generated data is effective for model selection. This

<sup>4</sup>We focus on only SQuAD here because our other two datasets have small or nonexistent training splits.

<sup>5</sup>The closest dataset retrieved by the dataset retriever for our SQuAD-inspired prompt is The Children’s Book Test Dataset (Hill et al., 2016).

| Method                      | #Train | Performance | Anno. Cost |
|-----------------------------|--------|-------------|------------|
| Retrieval only              | 3,000  | 56.79       | ≈ \$ 0     |
| Generation only             | 3,000  | 44.20       | ≈ \$ 5     |
| <b>Retrieval+generation</b> | 6,000  | 61.46       | ≈ \$ 5     |
| Custom annotation           | 3,000  | 61.64       | ≈ \$ 540   |

Table 2: We compare performance on SQuAD on an annotation-cost basis, using datasets produced by different modules of Prompt2Model, along with manual annotation. We use exact match accuracy on the real SQuAD test set<sup>6</sup> to measure *the true task performance*. The cost of custom annotation is calculated using the reported annotator pay rate of \$9/hour from Rajpurkar et al. (2016) and keeping 1,000 validation examples.

| Dataset      | Metric | $\tau$ | $p$ -value |
|--------------|--------|--------|------------|
| SQuAD        | EM     | 64.3   | 0.03*      |
| Temporal     | ChrF++ | 24.2   | 0.31       |
| MCoNaLa (JP) | ChrF++ | 70.9   | 0.00**     |

Table 3: We evaluate 10 different models on real test sets and their corresponding generated clones. We compute Kendall’s Tau on the ranked lists of models and find statistically significant correlations for 2 of 3 datasets.

is closely related to the concept of concurrence between benchmarks (Liu et al., 2023a).

Table 3 shows Kendall’s  $\tau$  for each task, computed over a set of reasonable models.<sup>7</sup> The generated data shows a strong correlation to the true performance on two of the three datasets.

## 6 Discussion and Conclusion

We propose Prompt2Model, a framework that automatically constructs task-specific models using only natural language prompts. Our proof-of-concept experiments show that, despite using a similar simple interface like LLMs, Prompt2Model delivers small yet accurate models and its generated datasets can be used to estimate real-world performance. Prompt2Model’s extensible and modular design makes it a platform for advancing model distillation, dataset generation, synthetic evaluation, dataset retrieval, and model retrieval.

We believe our Prompt2Model framework can inspire various novel research questions. These questions may include how much data should we generate for downstream model training and how diverse should it be? How do we effectively mix retrieved and generated data to achieve comple-

<sup>7</sup>This set of models consisted of 5 T5-family models, 2 BART-family models, and 1-5 additional retrieved models from the *Model Retriever*, depending on the task.

mentary strengths (e.g. using dataset generation to focus on inputs that the retrieved dataset fails to cover)? Since users may struggle to articulate their needs, future extensions should address human-in-the-loop correction by enabling either iterative editing of prompts or post-hoc fixing of data and models when the task metadata extraction and generated data do not align with user intentions. We invite the community to contribute novel implementations of various components in our framework.

## Limitations

Our paper uses proprietary LLM APIs in our experiments, which is problematic as a scientific artifact (Rogers et al., 2023). Our software supports open-source LLMs to avoid reliance on proprietary APIs.

Another limitation of our work is the limited ability of Prompt2Model to support tasks that require processing languages other than English. While we have shown the limitations of our system at supporting code generation from Japanese natural language queries, our system is likely to struggle more with lower-resource languages. We use the unpublished gpt-3.5-turbo model for our Dataset Generator in our reference implementation. This model is believed to be similar to GPT-3 (Brown et al., 2020), which was trained on 93% English documents. Our use of this model may exacerbate existing disparities between high-resource languages and low-resource languages.

One potential limitation is that we have only tested our approach on 3 tasks, each with a single dataset and a single evaluation metric. We justify this decision because our focus is on providing an extensible software system rather than establishing state-of-the-art results on many datasets, but we believe that our results suggest broader applicability.

## Acknowledgements

VV was supported by a fellowship from NEC Laboratories Europe. CZ was supported in part by the Deng Feng Fund from the Beijing National Research Center for Information Science and Technology. We are grateful to Alex Cabrera, Will Epperson, Nelson Liu, Arjun Ramani, Zirui Cheng, Zhiyuan Zeng, Tianci Xue, Yanchen Liu, Yi-Hsin Hung and Zhilin Yang for their guidance. We particularly appreciate Zirui Cheng’s video production support for our demo.

## Ethics Statement

Any system which makes powerful technology more accessible to the public has ethical implications. [Widder et al. \(2022\)](#) discuss ethical issues with open-source packages in relation to software libraries for deepfaking, including the possibility of enabling malicious actors to use technology that they would otherwise not have the technical skills to leverage. This is also a risk for an AutoML system such as Prompt2Model; however, we believe this risk is outweighed by the benefits of greater accessibility, especially given that a low barrier to entry for generating harmful data already exists in the form of prompted, web-interface models.

While Prompt2Model could, if given harmful inputs, generate toxic, offensive, or inaccurate synthetic data, this is no more of a risk with Prompt2Model than it is with the underlying prompted model ([Bender et al., 2021](#)); indeed, the use of models and supplementary datasets retrieved from Hugging Face may lessen the likelihood of a downstream model replicating harms from the prompted model’s outputs, though more investigation is needed. Like all ML models, the models that Prompt2Model returns can make mistakes, and we aim to be transparent in our documentation about potential limitations of the system.

We hope that Prompt2Model will be broadly useful. Our work is motivated by a desire to increase the accessibility of NLP models to people who are not in the NLP community but would benefit from the community’s innovations; particularly, to people who would use NLP models downstream but may not have the domain-specific knowledge to design their system. Prompt2Model may also prove useful for early NLP researchers by providing a starting point for intuitions about baselines for various tasks and enabling the discovery of similarities between a described task and existing work. We open-source Prompt2Model and welcome community contributions.

## 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*.

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*, FAccT ’21, page 610–623, New York, NY, USA. Association for Computing Machinery.

- Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew R. Gormley. 2023. [It’s MBR all the way down: Modern generation techniques through the lens of minimum bayes risk](#). In *EMNLP 2023 Big Picture Workshop*, Singapore.
- 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.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Nitay Calderon, Subhabrata Mukherjee, Roi Reichart, and Amir Kantor. 2023. [A systematic study of knowledge distillation for natural language generation with pseudo-target training](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14632–14659, Toronto, Canada. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Michael Färber and Ann-Kathrin Leisinger. 2021. Recommending datasets for scientific problem descriptions. In *CIKM*, pages 3014–3018.
- Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.
- Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. [Precise zero-shot dense retrieval without relevance labels](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

- Arnav Gudibande, Eric Wallace, Charles Burton Snell, Xinyang Geng, Hao Liu, P. Abbeel, Sergey Levine, and Dawn Song. 2023. [The false promise of imitating proprietary llms](#). *ArXiv*, abs/2305.15717.
- Xingwei He, Zheng-Wen Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. [Annollm: Making large language models to be better crowdsourced annotators](#). *ArXiv*, abs/2303.16854.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. [The goldilocks principle: Reading children’s books with explicit memory representations](#).
- Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. [Promptmaker: Prompt-based prototyping with large language models](#). In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–8.
- Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.
- 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, Guntan 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.
- Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2023a. [Do question answering modeling improvements hold across benchmarks?](#) In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13186–13218, Toronto, Canada. Association for Computational Linguistics.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. [Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing](#). *ACM Comput. Surv.*, 55(9).
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. [Rethinking the role of demonstrations: What makes in-context learning work?](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Andrei Paleyey, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. [Challenges in deploying machine learning: a survey of case studies](#). *ACM Computing Surveys*, 55(6):1–29.
- Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2022. [Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models](#).
- 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, Lisbon, Portugal. 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(1).
- 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.
- Laria Reynolds and Kyle McDonell. 2021. [Prompt programming for large language models: Beyond the few-shot paradigm](#). In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA ’21, New York, NY, USA. Association for Computing Machinery.
- Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. [Okapi at trec-4](#). In *Text Retrieval Conference*.
- Anna Rogers, Niranjan Balasubramanian, Leon Derczynski, Jesse Dodge, Alexander Koller, Sasha Lucioni, Maarten Sap, Roy Schwartz, Noah A Smith, and Emma Strubell. 2023. [Closed ai models make bad baselines](#).
- Vijay Viswanathan, Luyu Gao, Tongshuang Wu, Pengfei Liu, and Graham Neubig. 2023. [DataFinder: Scientific dataset recommendation from natural language descriptions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10288–10303, Toronto, Canada. Association for Computational Linguistics.
- Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. [Want to reduce labeling cost? GPT-3 can help](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.



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.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. 2023. *MCoNaLa: A benchmark for code generation from multiple natural languages*. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 265–273, Dubrovnik, Croatia. Association for Computational Linguistics.

David Gray Widder, Dawn Nafus, Laura Dabbish, and James Herbsleb. 2022. *Limits and possibilities for “ethical ai” in open source: A study of deepfakes*. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT ’22*, page 2035–2046, New York, NY, USA. Association for Computing Machinery.

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.

Sherry Wu, Hua Shen, Daniel S Weld, Jeffrey Heer, and Marco Tulio Ribeiro. 2023. *Scattershot: Interactive in-context example curation for text transformation*. In *Proceedings of the 28th International Conference on Intelligent User Interfaces, IUI ’23*, page 353–367, New York, NY, USA. Association for Computing Machinery.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. *Bertscore: Evaluating text generation with bert*.

# NEWSSENSE: Reference-free Verification via Cross-document Comparison

Jeremiah Milbauer\* and Ziqi Ding† and Zhijin Wu† and Tongshuang Wu

Carnegie Mellon University, Pittsburgh PA, USA

{ jmilbaue | sherryw }@cs.cmu.edu

{ ziqiding | zhijinw }@andrew.cmu.edu

## Abstract

We present NEWSSENSE, a novel sensemaking tool and reading interface designed to collect and **integrate** information from multiple news articles on a central topic. NEWSSENSE provides “reference-free verification,” augmenting a central grounding article of the user’s choice by: (1) linking it to related articles from different sources; and (2) providing inline highlights on how specific claims are either supported or contradicted by information from other articles. Using NEWSSENSE, users can seamlessly digest and cross-check multiple information sources without disturbing their natural reading flow. Our pilot study shows that NEWSSENSE has the potential to help users identify key information, verify the credibility of news articles, explore different perspectives, and understand what content is supported, contradicted, or missing. NEWSSENSE is available open source at [github.com/jmilbauer/NewsSense](https://github.com/jmilbauer/NewsSense).

## 1 Introduction

Why is it so hard, and so exhausting, to read the news? In the quest for knowledge, news readers today must contend with a rapidly evolving 24-hour news cycle, multiple news venues competing for attention and clicks, and the challenge of integrating fact-based reporting, opinion pieces, and social media commentary (Lazer et al., 2018; Benkler et al., 2018; Farkas and Schou, 2019). With news becoming increasingly politicized (Faris et al., 2017) readers also face the challenge of identifying and avoiding misinformation, disinformation, and hyperbolic “clickbait” as they try to remain informed about the world around them.

Various solutions have been proposed to assist with users’ news reading. For example, media watchdog companies have created media bias charts to represent political leaning and credibil-

\* Corresponding author.

† Equal contribution.

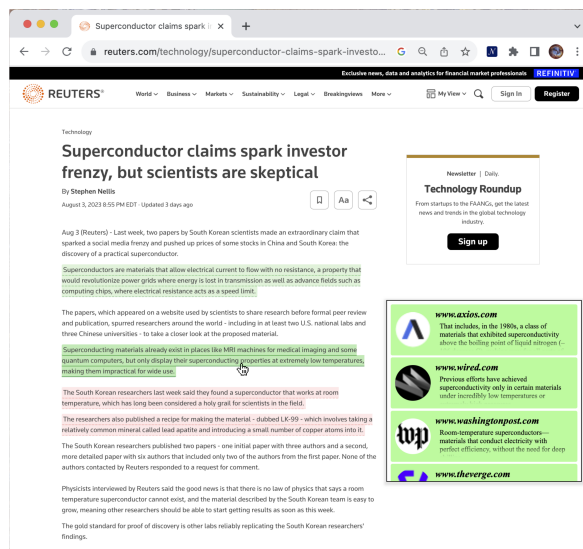


Figure 1: A screenshot from NEWSSENSE browser extension running in Chrome. The extension provides highlights indicated supported and controversial information. When the user clicks on a highlighted sentence, NEWSSENSE adds an scrollable overlay box containing snippets of external evidence.

ity of news sources<sup>1 2</sup>. However, these resources force users to rely on potentially untrustworthy third-party designations of media bias, which treat each *news source* as a whole, without digging into specific articles or topics.

While novel automatic fact checking (Thorne et al., 2018) and fake news detection (Zhou and Zafarani, 2020; Chen et al., 2015) systems can provide verification per-article, these approaches typically rely on a preordained corpus of verified facts which cannot keep up with the always-evolving facts, and may not reflect user preferences or multilateral perspectives. Aggregating articles from heterogeneous sources seem a more promising direction for cross-checking new facts (without predetermined groundtruths) and collecting different perspectives, but existing attempts are still too coarse and over-

<sup>1</sup><https://www.allsides.com/media-bias/media-bias-rating-methods>

<sup>2</sup><https://adfontesmedia.com/interactive-media-bias-chart/>

whelming. For example, both Google News' "Stories" <sup>3</sup> feature and Ground.news<sup>4</sup> collect articles about the same events but display them in the form of exhaustive lists – users are still forced to read and compare each article on its own.

We argue that instead of simply collecting and **aggregating** news articles, information and claims from multiple sources should be **integrated** in a way that allows users to identify fine-grained claim-level bias, spin, controversy, or evidence.

We present NEWSSENSE, a novel framework for sensemaking within a cluster of documents, to address the three key problems of news reading – bias, factuality, and article overload – in a single streamlined interface. NEWSSENSE leverages existing modular natural language processing (NLP) techniques to identify and link claims made across *a cluster of news articles*, such that these articles become references for each other. NEWSSENSE also displays the linking information using an *interactive reading interface*, which allows users to easily explore the cross-document connections without being overwhelmed. In pilot user studies, we see that the NEWSSENSE framework has the potential to help users identify key information, verify the credibility of news articles, and explore different perspectives.

While NEWSSENSE is primarily implemented for news articles, our framework can be easily generalized to other assisted reading and cross-checking scenarios (e.g., compare multiple manuscripts in literature reviews). The key contributions of NEWSSENSE are:

1. A pipeline for analyzing the connections between a collection of documents.
2. A two-stage method for efficiently computing cross-document links between claims that support or contradict each other, enabling "reference-free" fact checking.
3. A framework for visualizing cross-document connections, and integrating claims from multiple documents into a single reading experience.

We conclude by discussing the generality and potential social benefits of NEWSSENSE.

## 2 Related Work

This section covers related research across media analytics, sensemaking, and natural language processing. Though some core ideas of this work have

<sup>3</sup><https://news.google.com/stories/>

<sup>4</sup><https://ground.news>

been explored in the past, to our knowledge they have never been combined in a single system.

**Media Bias and Analytics** Research on media bias includes academic research to study social media sharing patterns (Roberts et al., 2021; Bakshy et al., 2015) and bias within media publications (Flaxman et al., 2016; Hamborg et al., 2019; Groseclose and Milyo, 2005). Commercial products exist in this area as well, such as the media bias charts of AllSides <sup>5</sup>, which classifies political slant into one of five categories, and Ad Fontes Media <sup>6</sup>, which models both political slant and factual credibility.

Research on news and social content aggregation has focused primarily on headline detection, timeline construction and clustering (Bouras and Tsogkas, 2012; Laban and Hearst, 2017), and event detection (Atefeh and Khreich, 2015; Kumaran and Allan, 2004). There exist user-oriented products in this space, such as Google News Stories <sup>7</sup>, and Ground.news <sup>8</sup>. Some outlets, such as ProPublica, aggregate their news stories into timelines <sup>9</sup>.

**Reading Interfaces and Sensemaking** Recent work on reading interfaces has primarily focused on scientific literature, augmenting documents with information about cited papers (Lo et al., 2023; Kang et al., 2022), or enhancing references within documents themselves (Head et al., 2021; Liu et al., 2023a, 2019a, 2023b).

For the News domain specifically, Laban and Hearst (2017) aggregates articles and extracts key quotes to construct a timeline for a given story. We are also aware of an abstract describing work to combine multiple article headlines and ledes into a single digestible form, though no follow-up is available (Glassman et al., 2020).

**Fact Verification and NLI** Natural Language Inference is a task focused on classifying the relationship between a pair of sentences as either "neutral", "entailment", or "contradiction." Datasets such as SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2017) have become major benchmarks for natural language processing research. Recent work has also considered document-level NLI (Koreeda and Manning, 2021; Chen et al., 2022), as well

<sup>5</sup><https://www.allsides.com/media-bias/media-bias-rating-methods>

<sup>6</sup><https://adfontesmedia.com/interactive-media-bias-chart/>

<sup>7</sup><https://news.google.com>

<sup>8</sup><https://ground.news/>

<sup>9</sup><https://www.propublica.org/series>

as cross-document reasoning based in NLI (Schuster et al., 2022), and scalable pairwise reasoning (Milbauer et al., 2023).

There is also a growing body of work on NLP systems for fact verification and attribution. Recent datasets include FEVER (Thorne et al., 2018) and VitaminC (Schuster et al., 2021), as well as datasets focused on real-world examples of updating, editing, and citing claims in domains like news and Wikipedia (Petroni et al., 2022; Spangher et al., 2022; Iv et al., 2022).

### 3 The NEWSSENSE Framework

The core philosophy behind NEWSSENSE is to go beyond article aggregation by integrating the information contained within a cluster of related news articles into the reading experience.

NEWSSENSE starts with a single “focus” article and a set of related “background” articles. The distinction between focus and background article is arbitrary, as any article within the cluster could be designated the focus article. NEWSSENSE then identifies claims within the focus article that are related – either by contradiction or entailment – to claims within the background articles. The claims in the focus article are then highlighted, and linked to the background articles so that users can explore the supporting or contradicting evidence for a given claim without just relying on third-party measurements of bias or credibility.

The NEWSSENSE interface has three primary components: a Focus Article, Sentence Highlights, and External Evidence. Together, these elements display the computed connections between the focus article and the background articles.

#### 3.1 Focus Article

The NEWSSENSE interface features a central panel that displays the focus article, including the entire news article the reader is interested in primarily. The focus article can be presented through a dedicated application, or by adding NEWSSENSE as an overlay on top of the existing web browsing experience.

#### 3.2 Sentence Highlights

The interface highlights claims made in the focus article that have supporting articles with green, while claims with contradicting articles are highlighted with red. By doing so, readers can quickly and easily identify areas of agreement and disagree-

ment across different news sources. However, not all disagreement is conflict; articles may disagree because one simply has more updated information than the other. Disagreements may also be limited to some individual claims, while the content of two articles remains generally consistent. This is one reason why it is essential to link to the external evidence, and allow the user to explore.

#### 3.3 External Evidence

When the reader hovers or clicks on a highlighted claim, an overlay panel appears, containing the supporting or contradicting claim excerpts, as well as their news sources. Although we do not implement it here, NEWSSENSE provides the opportunity to integrate insights from prior analyses of the news-media landscape by annotating links with metadata about the credibility or political slant of the referenced news venue.

For readers’ convenience, each supporting or contradicting claim is clickable and directs the reader to the origin of the associated claim. This allows readers to quickly access the relevant claims without having to search through an entire article, and helps them better understand the circumstances of the agreement or the disagreement. For a standalone NEWSSENSE interface not embedded in a web browser, readers would be prompted with a “back” button in the secondary articles to quickly go back to the focus article. By providing this functionality, readers can easily navigate through the focus and secondary articles and compare viewpoints, further enhancing their understanding of the news story.

### 4 Pilot Study

To gather feedback on the proposed NEWSSENSE interface and provide insights for the actual implementation, we conducted a pilot user study using a NewsReader mockup built with Figma<sup>10</sup>. This section describes the design and results of the study.

#### 4.1 Study Design

We aim to collect feedback on NEWSSENSE’s basic functionality, interface design, and content quality.

The participants were assigned a task of reading a news article using NEWSSENSE and answering a set of questions. The questions focused on the content of the news, how and where the user located information, and their level of trust in the

<sup>10</sup><https://www.figma.com/>

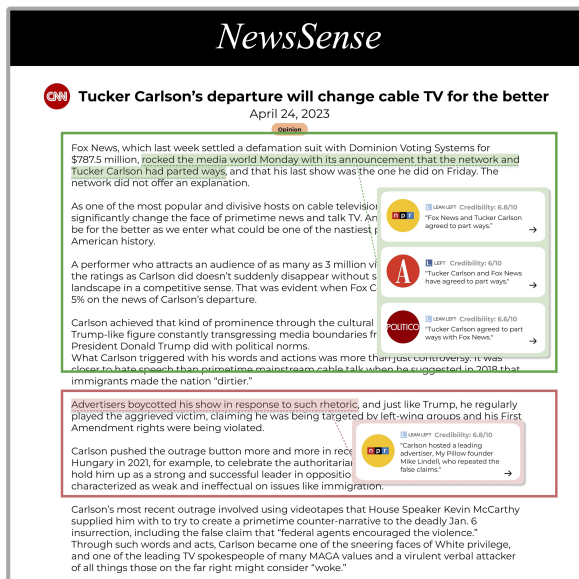


Figure 2: The design layout of the pilot study, prototyped in Figma. The article is presented in a central panel, featuring claims with supporting articles highlighted in green (section boxed in green), and claims with contradicting articles highlighted in red (section boxed in red). Each claim has an associated overlay box of external evidence that appears when the user hovers over the text.

information. These questions aimed to assess the basic functionality of NEWSSENSE in helping readers understand news comprehensively, to motivate further development of the system.

## 4.2 Results

Following the pilot user study with over 10 users, we identified several key findings. First, all users found NEWSSENSE to be useful in locating important information and verifying the credibility of news articles, aligning with our initial goal. The user-friendly interface of NEWSSENSE was well-received, though participants suggested enhancing interactivity to set it apart from other solutions. For instance, displaying real-time feedback like *"NewsSense is analyzing the article"* during loading.

Regarding content quality, some users found NEWSSENSE limited and suggested increased labeling or categorization within articles. One user noted *"Two highlighted sentences per page are insufficient for in-depth analysis."* User preferences varied for article summarization, with some wanting more key points and others preferring brevity. Contradicting previous feedback, one user preferred *"Summarizing key points only, rather than selecting sentences with unclear relevance."* Addressing this,

NEWSSENSE could allow customization, letting users choose key point count and filter supporting/contradicting data.

## 4.3 Study Takeaways

We found that users liked how NEWSSENSE highlighted important sentences from an article. We realized that the claims which are consistent across multiple articles (ie, those which are supported at least once) are likely to be the most important aspects to a given story. NEWSSENSE could thus help inform readers when there are key claims from across the article cluster missing from the article they are reading.

We also found that the bias labels for news venues could be overwhelming, and including them ran counter to our aim of reference-free verification; we eliminated these labels.

Users also appreciated how highlighted sentences functioned as summaries. Consequently we enhance the visibility of text highlights and further emphasize the alignment or contradiction of specific source by making the External Evidence cards colored accordingly.

## 5 System Overview

Following our user study, we implemented the NEWSSENSE framework as a browser plugin, which adds augmentations to news articles encountered on the web. Figure 1 shows the final appearance of the browser plugin. Open source code for the system and plugin, as well as a demo video, can be accessed at [github.com/jmilbauer/NewsSense](https://github.com/jmilbauer/NewsSense).

This section provides a description of the natural language processing system which powers NEWSSENSE. Figure 3 illustrates the four general steps of the pipeline: Collection, Selection, Filtering, and Linking.

### 5.1 Article Collection

First, we must collect a cluster of news articles that are all about the same news event. Our implementation scrapes data from Google News Stories, a website that collects many articles about the same events across news venues. After collecting article URLs via Google News Stories, we then collect the content of each article. A typical story contains over 50 articles.

### 5.2 Claim Selection

The next phase of the pipeline is to select the claims within each article cluster. We initially assumed

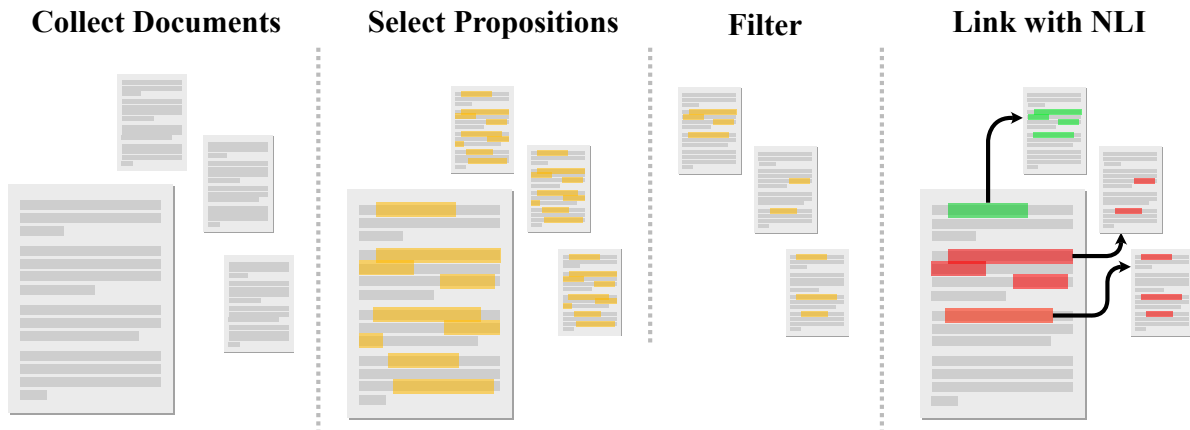


Figure 3: The four stages of the NEWSREADER linking pipeline: Article collection, Claim detection, claim filtering, and claim linking

a 1-to-1 mapping between sentences and claims, but quickly found that news articles often contain complex multi-clause sentences, which are not suitable for natural language inference. To address this issue, we few-shot prompting to generate a list of claims from sentences using a large language model (LLM). In our experiments, prompt exemplars are drawn from the PROPSEGMENT dataset (Chen et al., 2022), and the LLM used is OpenAI text-davinci-003. Full prompt details are provided in Appendix A. We also note that the authors of PROPSEGMENT report that T5-Large performs reasonably well on a similar proposition-level task, suggesting the possibility for further pipeline improvements, or one that does not rely on APIs.

### 5.3 Claim Filtering

Articles often contain over 30 sentences. For a cluster of 50 articles, a pairwise comparison of the full cartesian product of sentences has  $\mathcal{O}((NL)^2)$ , which is in practice well over 1,000,000 comparisons. Performing this level of computation at scale, even if we are pre-computing results for each article cluster, is simply not feasible. To address this, we perform an initial filtering step with leverages the fact that the vast majority of claims across any two articles are unrelated. We consider two approaches for claim filtering: Embedding Similarity filtering (ES) and Lexical Overlap filtering (LeO).

For Embedding Similarity filtering, we encode each claim in each article using a Transformer-based sentence encoder. Then, for each claim we retain only the  $k$  most similar other claims for comparison. In our implementation, we use the Sentence Transformers (Reimers and Gurevych, 2019)

|                      | Prec. | Rec. | Macro-F1 | TNR  |
|----------------------|-------|------|----------|------|
| Embedding Similarity | 0.99  | 0.95 | 0.97     | 0.99 |
| Lexical Overlap      | 0.91  | 0.89 | 0.90     | 0.91 |

Table 1: The positive-class precision, recall, macro-averaged F1, and true negative rate for the two filtering methods. Embedding Similarity outperforms lexical overlap on every metric.

model all-Mini-LM-L6-v2.

For Lexical Overlap filtering, we compare each sentence only with sentences that have overlapping words, as these sentences are likely to discuss similar topics. In our implementation, we process claims by first remove stopwords, then stemming using the NLTK (Loper and Bird, 2002) implementation of the Porter Stemmer (Porter, 1980), and compute overlap scores using the Jaccard Index.

We evaluated each filtering method on the MNLI (Williams et al., 2017) validation data, treating pairs of randomly sampled sentences as negative examples, and labeled “entailment” and “contradiction” sentence pairs as positive examples. For ES, we set a threshold of 0.3 cosine similarity; for LeO, we set a threshold of 0.1 overlap. We note that all-Mini-LM-L6-v2 included MNLI in its training data.

We include a summary of the results of these experiments in Table 1, which indicates that the ES method outperforms the LeO method. Of particular interest is the true negative rate, as this indicates the percentage of non-related sentences we expect to filter out.

## 5.4 Claim Linking

Once claim pairs have been filtered, we classify each pair according to the Natural Language Inference (NLI) framework, as “entailment,” “contradiction,” or “neutral.” We employ a pretrained language model, RoBERTa (Liu et al., 2019b), which was then fine-tuned on MNLI (Williams et al., 2017), a popular dataset for NLI. We download this fine-tuned version of RoBERTa from the Hugging Face model library <sup>11</sup>.

To avoid clutter, we keep fewer than 100 of the most confident predictions for each positive class (entailment or contradiction) within the article cluster. Claims are then assigned back to the sentences from which they were generated, and the sentence pairs are linked.

## 6 Discussion

NEWSSENSE provides an intuitive and effective interface for integrating information from a large cluster of news articles into a single, focused reading experience. Although applied in this demo to news articles, the NEWSSENSE framework could just as easily be applied to the analysis of other types of document clusters as well. The pipeline itself is highly modular, and can easily adopt advancements in NLP technologies to increase the accuracy or decrease processing time.

### 6.1 Future Work

The generality of the NEWSSENSE also introduces a number of opportunities for future development.

**Expanding the Scope of NewsReader** Often, articles contain references to past events. In the future, we would like to explore the possibility of extending the NEWSSENSE framework beyond the immediately article clusters to include all relevant articles in a timeline of events.

Additionally, as we explored the NEWSSENSE framework, we noticed that the clustering approach we used – the Google News Stories – sometimes established associations between source *news* articles, and background *primary source* articles. As a result, we would encourage further exploration of the NEWSSENSE framework when applied to heterogeneous and primary-source document collections, which might include primary scholarly literature.

<sup>11</sup><https://huggingface.co/roberta-large-mnli>

Beyond the news, we also anticipate that the NEWSSENSE framework could prove useful for the analysis of scientific or other scholarly articles, by tracing ideas and providing attribution across documents.

**NLP Pipeline Improvements** Because the NEWSSENSE pipeline is modular, a number of improvements can be explored. For sentence segmentation, methods that use a fine-tuned language model could improve the speed of segmentation. For sentence filtering, we relied on a pretrained sentence retrieval model – but future iterations of NEWSSENSE could use a sentence retrieval model fine-tuned on “unrelated” pairs of MNLI sentences.

We also found that in many cases, the NLI algorithm used for claim linking made mistakes, perhaps owing to the fact that news articles may not perfectly match models trained on MNLI. To address this, more robust approaches to NLI such as VITAMINC (Schuster et al., 2021) could be explored. Other NLI approaches that are particularly applicable to NEWSSENSE would be those designed for both premises and sentences, such as SeNtLI (Schuster et al., 2022), or for many-to-many reasoning, such as LAIT (Milbauer et al., 2023), which speeds up inference through late interaction. Depending on how NEWSREADER is deployed, models of different sizes could be used: larger state of the art models for a centralized server, or smaller models for performing fast inference on the user’s device.

**More Useful Information** Our final version of NEWSSENSE focused on a relatively paired-down and streamlined interface. However, users did suggest that they would like to see article summaries, and in fact used the highlighted claims as pseudo-summaries – indeed, good information is likely repeated across multiple articles. We would consider adding a way for NEWSSENSE to gather and convey the highlights – the key, supported, claims from across the article cluster – when a user is reading an article. We noticed other forms of unintended but incredibly useful functionality: For example, as stories develop, new facts emerge that may contradict old ones. This means that newer articles might supersede older ones. Future iterations of NEWSSENSE should help readers understand when a contradiction may be due to evolving stories.

**Deployment** A larger-scale user study would help determine what further improvements could

be made to the framework. Our fully interactive interface would help us run a study at larger scale.

## 7 Conclusion

We presented a novel framework for sensemaking within a cluster of documents. We applied this framework to news articles, building NEWSSENSE, an interactive tool that links claims within one document to supporting or contradicting evidence across the entire document cluster. NEWSSENSE assists readers by helping them to understand the connections and perspectives across many documents. Readers can thus attain a more comprehensive understanding of a given subject, while avoiding the dangers of information overload. Crucially, NEWSSENSE provides a framework for *reference-free* fact verification, which is essential in domains such as the news where events evolve in real time, because a knowledge source for factual grounding may not be available.

Our work expands the growing body of literature on natural language processing applications to document-level sensemaking by demonstrating the utility of automatically generated cross-document links, as well as the application of sensemaking tools to the news reading experience.

## Limitations and Ethics

NEWSSENSE falls within the genre of computer science literature that aims to solve problems such as misinformation. A broad critique of this literature is that it may be considered a form of *techno-solutionism*, in the sense that we seek to develop technological solutions to problems that are potentially social in origin, and perhaps better addressed with a social approach. However, we posit that because the problem of misinformation propagation and newsmedia overload are both enabled by technology, we do have a responsibility to explore the ability of technological systems to address these challenges. Unlike techniques that involve traditional fact verification, the reference-free approach of NEWSSENSE does not take on the role of deciding what is true and what is not; it simply helps users understand the context of each claim, and make their own decisions.

Beyond this broad critique, NEWSSENSE might be limited in its application: it presupposes a diverse and free media environment, it does not distinguish between venue quality, might suffer from false balance, and could give undue credence to

ideas repeated across low-quality outlets. These issues could result in NEWSSENSE providing a false sense of factuality. Additionally, there may be obstacles to its adoption, as the people who choose to use a system such as NEWSSENSE may already be predisposed to consider and critically evaluate diverse perspectives in the news; not those who need it most. We also consider that the highlighted links may clutter the reading experience, but we believe this concern is mitigated by the fact that news websites are already quite cluttered (by ads, sponsored links, and article thumbnails) and that users found the highlights helpful in identifying the key components of the articles.

## Acknowledgements

We thank the members of the Human-Centered Natural Language Processing course at Carnegie Mellon University for their comments and suggestions. We would also like to thank Tal Schuster and Emma Strubell for discussions that helped improve this work. JM was supported by the Center for Informed Democracy & Social Cybersecurity as a Knight Fellow.

## References

- Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164.
- Eytan Bakshy, Solomon Messing, and Lada A Adamic. 2015. Exposure to ideologically diverse news and opinion on facebook. *Science*, 348(6239):1130–1132.
- Yochai Benkler, Robert Faris, and Hal Roberts. 2018. *Network propaganda: Manipulation, disinformation, and radicalization in American politics*. Oxford University Press.
- Christos Bouras and Vassilis Tsogkas. 2012. A clustering technique for news articles using wordnet. *Knowledge-Based Systems*, 36:115–128.
- Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.
- Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, Dan Roth, and Tal Schuster. 2022. Propsegment: A large-scale corpus for proposition-level segmentation and entailment recognition. *arXiv preprint arXiv:2212.10750*.
- Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. Misleading online content: recognizing clickbait as"



- false news". In *Proceedings of the 2015 ACM on workshop on multimodal deception detection*, pages 15–19.
- Robert Faris, Hal Roberts, Bruce Etling, Nikki Bourassa, Ethan Zuckerman, and Yochai Benkler. 2017. Partisanship, propaganda, and disinformation: Online media and the 2016 us presidential election. *Berkman Klein Center Research Publication*, 6.
- Johan Farkas and Jannick Schou. 2019. *Post-truth, fake news and democracy: Mapping the politics of falsehood*. Routledge.
- Seth Flaxman, Sharad Goel, and Justin M Rao. 2016. Filter bubbles, echo chambers, and online news consumption. *Public opinion quarterly*, 80(S1):298–320.
- Elena L. Glassman, Janet Sung, Katherine Qian, Yuri Vishnevsky, and Amy Zhang. 2020. Triangulating the news: Visualizing commonality and variation across many news stories on the same event.
- Tim Groseclose and Jeffrey Milyo. 2005. A measure of media bias. *The quarterly journal of economics*, 120(4):1191–1237.
- Felix Hamborg, Karsten Donnay, and Bela Gipp. 2019. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries*, 20(4):391–415.
- 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*, CHI ’21, New York, NY, USA. Association for Computing Machinery.
- Robert Iv, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2022. Fruit: Faithfully reflecting updated information in text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3670–3686.
- Hyeonsu Kang, Joseph Chee Chang, Yongsung Kim, and Aniket Kittur. 2022. [Threddy: An interactive system for personalized thread-based exploration and organization of scientific literature](#). In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST ’22, New York, NY, USA. Association for Computing Machinery.
- Yuta Koreeda and Christopher D Manning. 2021. Contractnli: A dataset for document-level natural language inference for contracts. *arXiv preprint arXiv:2110.01799*.
- Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304.
- Philippe Laban and Marti A Hearst. 2017. newslens: building and visualizing long-ranging news stories. In *Proceedings of the Events and Stories in the News Workshop*, pages 1–9.
- David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science*, 359(6380):1094–1096.
- Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019a. [Unakite: Scaffolding developers’ decision-making using the web](#). In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST ’19, page 67–80, New York, NY, USA. Association for Computing Machinery.
- Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A. Myers. 2023a. [Selenite: Scaffolding decision making with comprehensive overviews elicited from large language models](#).
- Nelson F Liu, Kenton Lee, and Kristina Toutanova. 2023b. Anchor prediction: Automatic refinement of internet links. *arXiv preprint arXiv:2305.14337*.
- 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*.
- Kyle Lo, Joseph Chee Chang, Andrew Head, Jonathan Bragg, Amy X. Zhang, Cassidy Trier, Chloe Anastasiades, Tal August, Russell Authur, Danielle Bragg, Erin Bransom, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Yen-Sung Chen, Evie Yu-Yen Cheng, Yvonne Chou, Doug Downey, Rob Evans, Raymond Fok, Fangzhou Hu, Regan Huff, Dongyeop Kang, Tae Soo Kim, Rodney Kinney, Aniket Kittur, Hyeonsu Kang, Egor Klevak, Bailey Kuehl, Michael Langan, Matt Latzke, Jaron Lochner, Kelsey MacMillan, Eric Marsh, Tyler Murray, Aakanksha Naik, Ngoc-Uyen Nguyen, Srishti Palani, Soya Park, Caroline Paulic, Napol Rachatasumrit, Smita Rao, Paul Sayre, Zejiang Shen, Pao Siangliulue, Luca Soldaini, Huy Tran, Madeleine van Zuylen, Lucy Lu Wang, Christopher Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Marti A. Hearst, and Daniel S. Weld. 2023. [The semantic reader project: Augmenting scholarly documents through ai-powered interactive reading interfaces](#).
- Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.
- Jeremiah Milbauer, Annie Louis, Mohammad Javad Hosseini, Alex Fabrikant, Donald Metzler, and Tal Schuster. 2023. [LAIT: Efficient multi-segment encoding in transformers with layer-adjustable interaction](#). In *Proceedings of the 61st Annual Meeting of*

*the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10251–10269, Toronto, Canada. Association for Computational Linguistics.

Fabio Petroni, Samuel Broscheit, Aleksandra Piktus, Patrick Lewis, Gautier Izacard, Lucas Hosseini, Jane Dwivedi-Yu, Maria Lomeli, Timo Schick, Pierre-Emmanuel Mazaré, et al. 2022. Improving wikipedia verifiability with ai. *arXiv preprint arXiv:2207.06220*.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Hal Roberts, Rahul Bhargava, Linas Valiukas, Dennis Jen, Momin M Malik, Cindy Sherman Bishop, Emily B Ndulue, Aashka Dave, Justin Clark, Bruce Etling, et al. 2021. Media cloud: Massive open source collection of global news on the open web.

Tal Schuster, Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, and Donald Metzler. 2022. Stretching sentence-pair nli models to reason over long documents and clusters. *arXiv preprint arXiv:2204.07447*.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin c! robust fact verification with contrastive evidence. *arXiv preprint arXiv:2103.08541*.

Alexander Spangher, Xiang Ren, Jonathan May, and Nanyun Peng. 2022. Newsdits: A news article revision dataset and a novel document-level reasoning challenge. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 127–157.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Xinyi Zhou and Reza Zafarani. 2020. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53(5):1–40.

## A Appendix: Prompt for Claim Extraction

This is the prompt used for claim extraction from news article sentences:

Extract all the claims from a sentence, ignoring extraneous words such as unimportant adverbs. A sentence may contain multiple claims. Each claim should be of the form <subject> <predicate> <object>, and should have the first occurrence of any pronouns replaced by their antecedents.

Sentence: "The 3rd and 4th stations all announced that they would be postponed, and the Monaco station was subsequently cancelled."

Claim: Monaco station was cancelled.

Claim: 4th stations announced they would be postponed.

Claim: The 3rd stations announced they would be postponed.

Claim: The 4th stations postponed.

Claim: The 3rd stations postponed.

Sentence: "Lewis Hamilton and Mercedes have once again confirmed themselves as drivers and constructors world champions."

Claim: Mercedes confirmed themselves as constructors world champions.

Claim: Lewis Hamilton confirmed themselves as drivers world champions.

Sentence: "Local organizers in East Palestine, Ohio on Monday said their activism has successfully pressured rail company Norfolk Southern to agree to a limited relocation plan for some residents affected by last month's train derailment, but added they have no intention of backing down from their demand for justice for thousands of people in the area who are struggling in the aftermath of the accident."

Claim: Local organizers said their activism has pressured rail company Norfolk Southern to agree to a limited relocation plan.

Claim: Local organizers have no intention of backing down from their demand for justice.

Claim: Rail company Norfolk Southern agree to a limited relocation plan.

Sentence: <INSERT SENTENCE HERE>

# NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails

Traian Rebedea\*, Razvan Dinu\*, Makesh Sreedhar, Christopher Parisien, Jonathan Cohen

NVIDIA

Santa Clara, CA

{trebedea, rdinu, makeshn, cparisien, jocohen}@nvidia.com

## Abstract

NeMo Guardrails is an open-source toolkit<sup>1</sup> for easily adding programmable guardrails to LLM-based conversational systems. Guardrails (or *rails* for short) are a specific way of controlling the output of an LLM, such as not talking about topics considered harmful, following a predefined dialogue path, using a particular language style, and more. There are several mechanisms that allow LLM providers and developers to add guardrails that are embedded into a specific model at training, e.g. using model alignment. Differently, using a runtime inspired from dialogue management, NeMo Guardrails allows developers to add *programmable* rails to LLM applications - these are user-defined, independent of the underlying LLM, and interpretable. Our initial results show that the proposed approach can be used with several LLM providers to develop controllable and safe LLM applications using programmable rails.

## 1 Introduction

Steerability and trustworthiness are key factors for deploying Large Language Models (LLMs) in production. Enabling these models to stay on track for multiple turns of a conversation is essential for developing task-oriented dialogue systems. This seems like a serious challenge as LLMs can be easily led into veering off-topic (Pang et al., 2023). At the same time, LLMs also tend to generate responses that are factually incorrect or completely fabricated (*hallucinations*) (Manakul et al., 2023; Peng et al., 2023; Azaria and Mitchell, 2023). In addition, they are vulnerable to prompt injection (or jailbreak) attacks, where malicious actors manipulate inputs to trick the model into producing harmful outputs (Kang et al., 2023; Wei et al., 2023; Zou et al., 2023).

Building trustworthy and controllable conversational systems is of vital importance for deploy-

\*Equal contribution

<sup>1</sup><https://github.com/NVIDIA/NeMo-Guardrails>

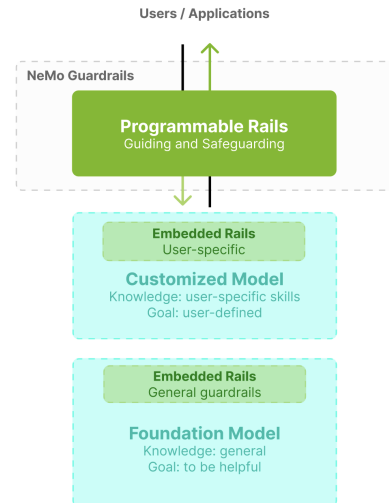


Figure 1: Programmable vs. embedded rails for LLMs.

ing LLMs in customer facing situations. NeMo Guardrails is an open-source toolkit for easily adding programmable rails to LLM-based applications. Guardrails (or *rails*) provide a mechanism for controlling the output of an LLM to respect some human-imposed constraints, e.g. not engaging in harmful topics, following a predefined dialogue path, adding specific responses to some user requests, using a particular language style, extracting structured data. To implement the various types of rails, several techniques can be used, including model alignment at training, prompt engineering and chain-of-thought (CoT), and adding a dialogue manager. While model alignment provides general rails embedded in the LLM at training and prompt tuning can offer user-specific rails embedded in a customized model, NeMo Guardrails allows users to define custom programmable rails at runtime as shown in Fig. 1. This mechanism is independent of alignment strategies and supplements embedded rails, works with different LLMs, and provides interpretable rails defined using a custom modeling language, Colang.

To implement user-defined programmable rails for LLMs, our toolkit uses a programmable runtime engine that acts like a proxy between the user and the LLM. This approach is complementary to model alignment and it defines the rules the LLM should follow in the interaction with the users. Thus, the Guardrails runtime has the role of a dialogue manager, being able to interpret and impose the rules defining the programmable rails. These rules are expressed using a modeling language called Colang. More specifically, Colang is used to define rules as dialogue flows that the LLM should always follow (see Fig. 2). Using a prompting technique with in-context learning and a specific form of CoT, we enable the LLM to generate the next steps that guide the conversation. Colang is then interpreted by the dialogue manager to apply the guardrails rules predefined by users or automatically generated by the LLM to guide the behavior of the LLM.

While NeMo Guardrails can be used to add safety and steerability to any LLM-based application, we consider that dialogue systems powered by an LLM benefit the most from using Colang and the Guardrails runtime. The toolkit is licensed as Apache 2.0, and we provide initial support for several LLM providers, together with starter example applications and evaluation tools.

## 2 Related Work

### 2.1 Model Alignment

Existing solutions for adding rails to LLMs rely heavily on model alignment techniques such as instruction-tuning (Wei et al., 2021) or reinforcement learning (Ouyang et al., 2022; Glaese et al., 2022; OpenAI, 2023). The alignment of LLMs works on several dimensions, mainly to improve helpfulness and to reduce harmfulness. Alignment in general, including red-teaming (Perez et al., 2022), requires a large collection of input prompts and responses that are manually labeled according to specific criteria (e.g., harmlessness).

Model alignment provides rails embedded at training in the LLM, that cannot easily be changed at runtime by users. Moreover, it also requires a large set of human-annotated response ratings for each rail to be incorporated by the LLM. While Reinforcement Learning from Human Feedback (Ouyang et al., 2022) is the most popular method for model alignment, alternatives such as RL from AI Feedback (Bai et al., 2022b) do not

```
define flow
  user express greeting
  bot express greeting

define flow
  user ask math question
  do ask wolfram alpha

define flow
  user ask distance
  do ask wolfram alpha

define subflow ask wolfram alpha
  # Generate the full query for Wolfram Alpha.
  $full_wolfram_query = ...
  $result = execute wolfram alpha request
              (query=$full_wolfram_query)

  bot respond with result
```

Figure 2: Dialogue flows defined in Colang: a simple greeting flow and two topical rail flows calling the custom action `wolfram alpha request` to respond to math and distance queries.

require a human labeled dataset and use the actual LLM to provide feedback for each response.

While most alignment methods provide general embedded rails, in a similar way developers can add app-specific embedded rails to an LLM via prompt tuning (Lester et al., 2021; Liu et al., 2022).

### 2.2 Prompting and Chain-of-Thought

The most common approach to add user-defined programmable rails to an LLM is to use prompting, including prompt engineering and in-context learning (Brown et al., 2020), by prepending or appending a specific text to the user input (Wang and Chang, 2022; Si et al., 2022). This text specifies the behavior that the LLM should adhere to.

The other approach to provide LLMs with user-defined runtime rails is to use chain-of-thought (CoT) (Wei et al., 2022). In its simplest form, CoT appends to the user instruction one or several similar examples of input and output pairs for the task at hand. Each of these examples contains a more detailed explanation in the output, useful for determining the final answer. Other more complex approaches involve several steps of prompting the LLM in a generic to specific way (Zhou et al., 2022) or even with entire dialogues with different roles similar to an inner monologue (Huang et al., 2022).

### 2.3 Task-Oriented Dialogue Agents

Building task-oriented dialogue agents generally requires two components: a Natural Language Understanding (NLU) and a Dialogue Management (DM) engine (Bocklisch et al., 2017; Liu et al.,

2021). There exist a wide range of tools and solutions for both NLU and DM, ranging from open-source solutions like Rasa (Bocklisch et al., 2017) to proprietary platforms, such as Microsoft LUIS or Google DialogFlow (Liu et al., 2021). Their functionality mostly follows these two steps: first the NLU extracts the intent and slots from the user message, then the DM predicts the next dialogue state given the current dialogue context.

The set of intents and dialogue states are finite and pre-defined by a conversation designer. The bot responses are also chosen from a closed set depending on the dialogue state. This approach allows to define specific dialogue flows that tightly control any dialogue agent. Conversely, these agents are rigid and require a high amount of human effort to design and update the NLU and dialogue flows.

At the other end of the spectrum are recent end-to-end (E2E) generative approaches that use LLMs for dialogue tracking and bot message generation (Hudeček and Dušek, 2023; Zhang et al., 2023). NeMo Guardrails also uses an E2E approach to build LLM-powered dialogue agents, but it combines a DM-like runtime able to interpret and maintain the state of dialogue flows written in Colang with a CoT-based approach to generate bot messages and even new dialogue flows using an LLM.

### 3 NeMo Guardrails

#### 3.1 General Architecture

NeMo Guardrails acts like a proxy between the user and the LLM as detailed in Fig. 3. It allows developers to define programmatic rails that the LLM should follow in the interaction with the users using **Colang**, a formal modeling language designed to specify flows of events, including conversations. Colang is interpreted by the **Guardrails runtime** which applies the user-defined rules or automatically generated rules by the LLM, as described next. These rules implement the guardrails and guide the behavior of the LLM.

An excerpt from a Colang script is shown in Fig. 2 - these scripts are at the core of a Guardrails app configuration. The main elements of a Colang script are: user canonical forms, dialogue flows, and bot canonical forms. All these three types of definitions are also indexed in a **vector database** (e.g., Annoy (Spotify), FAISS (Johnson et al., 2019)) to allow for efficient nearest-neighbors lookup when selecting the few-shot examples for the prompt. The interaction between the LLM and

the Guardrails runtime is defined using Colang rules. When prompted accordingly, the LLM is able to generate Colang-style code using few-shot in-prompt learning. Otherwise, the LLM works in normal mode and generates natural language.

**Canonical forms** (Sreedhar and Parisien, 2022) are a key mechanism used by Colang and the runtime engine. They are expressed in natural language (e.g., English) and encode the meaning of a message in a conversation, similar to an intent. The main difference between intents and canonical forms is that the former are designed as a closed set for a text classification task, while the latter are generated by an LLM and thus are not bound in any way, but are guided by the canonical forms defined by the Guardrails app. The set of canonical forms used to define the rails that guide the interaction is specified by the developer; these are used to select few-shot examples when generating the canonical form for a new user message.

Using these key concepts, developers can implement a variety of programmable rails. We have identified two main categories: topical rails and execution rails. **Topical rails** are intended for controlling the dialogue, e.g. to guide the response for specific topics or to implement complex dialogue policies. **Execution rails** call custom actions defined by the app developer; we will focus on a set of safety rails available to all Guardrails apps.

#### 3.2 Topical Rails

Topical rails employ the key mechanism used by NeMo Guardrails: Colang for describing programmable rails as **dialogue flows**, together with the Colang interpreter in the runtime for **dialogue management** (*Execute flow [Colang]* block in Fig. 3). Flows are specified by the developer to determine how the user conversation should proceed. The dialogue manager in the Guardrails runtime uses an event-driven design (an event loop that processes events and generates back other events) to ensure which flows are active in the current dialogue context.

The runtime has three main stages (see Fig. 3) for guiding the conversation with dialogue flows and thus ensuring the topical rails:

**Generate user canonical form.** Using similarity-based few-shot prompting, generate the canonical form for each user input, allowing the guardrails system to trigger any user-defined flows.

**Decide next steps and execute them.** Once the user canonical form is identified, there are two po-

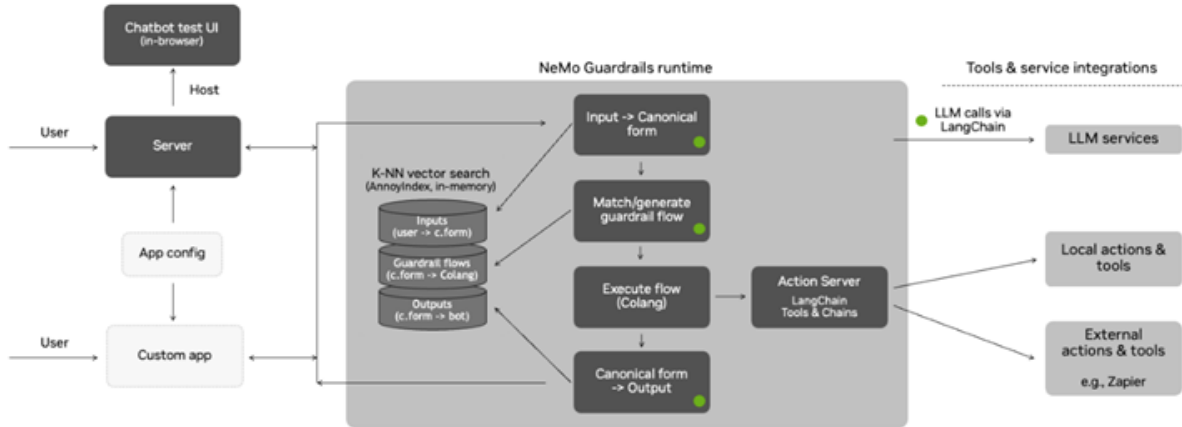


Figure 3: NeMo Guardrails general architecture.

tential paths: **1) Pre-defined flow:** If the canonical form matches any of the developer-specified flows, the next step is extracted from that particular flow by the dialogue manager; **2) LLM decides next steps:** For user canonical forms that are not defined in the current dialogue context, we use the generalization capability of the LLM to decide the appropriate next steps - e.g., for a travel reservation system, if a flow is defined for booking bus tickets, the LLM should generate a similar flow if the user wants to book a flight.

**Generate bot message(s).** Conditioned by the next step, the LLM is prompted to generate a response. Thus, if we do not want the bot to respond to political questions, and the next step for such a question is *bot inform cannot answer* – the bot would deflect from responding, respecting the rail.

Appendix B provides details about the Colang language. Appendix C contains sample prompts.

### 3.3 Execution Rails

The toolkit also makes it easy to add *“execution”* rails. These are custom actions (defined in Python), monitoring both the input and output of the LLM, and can be executed by the Guardrails runtime when encountered in a flow. While execution rails can be used for a wide range of tasks, we provide several rails for LLM safety covering fact-checking, hallucination, and moderation.

#### 3.3.1 Fact-Checking Rail

Operating under the assumption of retrieval augmented generation (Wang et al., 2023), we formulate the task as an entailment problem. Specifically, given an *evidence* text and a generated *bot response*,

we ask the LLM to predict whether the response is grounded in and entailed by the evidence. For each evidence-hypothesis pair, the model must respond with a binary entailment prediction using the following prompt:

*You are given a task to identify if the hypothesis is grounded and entailed in the evidence. You will only use the contents of the evidence and not rely on external knowledge. Answer with yes/no. "evidence": {{evidence}} "hypothesis": {{bot\_response}} "entails":*

If the model predicts that the hypothesis is not entailed by the evidence, this suggests the generated response may be incorrect. Different approaches can be used to handle such situations, such as abstaining from providing an answer.

#### 3.3.2 Hallucination Rail

For general-purpose questions that do not involve a retrieval component, we define a hallucination rail to help prevent the bot from making up facts. The rail uses self-consistency checking similar to SelfCheckGPT (Manakul et al., 2023): given a query, we first sample several answers from the LLM and then check if these different answers are in agreement. For hallucinated statements, repeated sampling is likely to produce responses that are not in agreement.

After we obtain  $n$  samples from the LLM for the same prompt, we concatenate  $n - 1$  responses to form the *context* and use the  $n^{\text{th}}$  response as the *hypothesis*. Then we use the LLM to detect if the sampled responses are consistent using the prompt template defined in Appendix D.

### 3.3.3 Moderation Rails

The moderation process in NeMo Guardrails contains two key components:

- **Input moderation**, also referred as *jailbreak* rail, aims to detect potentially malicious user messages before reaching the dialogue system.
- **Output moderation** aims to detect whether the LLM responses are legal, ethical, and not harmful prior to being returned to the user.

The moderation system functions as a pipeline, with the user message first passing through input moderation before reaching the dialogue system. After the dialogue system generates a response powered by an LLM, the output moderation rail is triggered. Only after passing both moderation rails, the response is returned to the user.

Both the input and output moderation rails are framed as another task to a powerful, well-aligned LLM that vets the input or response. The prompt templates for these rails are found in Appendix D.

## 4 Sample Guardrails Applications

Adding rails to conversation applications is simple and straightforward using Colang scripts.

### 4.1 Topical Rails

Topical rails can be used in combination with execution rails to decide when a specific action should be called or to define complex dialogue flows for building task oriented agents.

In the example presented in Fig. 2, we implement two topical rails that allow the Guardrails app to use the WolframAlpha engine to respond to math and distance queries. To achieve this, the `wolfram alpha request` custom action (implemented in Python, available on Github) is using the WolframAlpha API to get a response to the user query. This response is then used by the LLM to generate an answer in the context of the current conversation.

### 4.2 Execution Rails

The steps involved in adding executions rails are:

1. **Define the action** - Defining a rail requires the developer to define an action that specifies the logic for the rail (in Python).
2. **Invoke action in dialogue flows** - Once the action has been defined, we can call the action from Colang using the `execute` keyword.
3. **Use action output in dialogue flow** - The developer can specify how the application should react to the output from the action.

Appendix E contains details about defining actions, together with an example of the actions that implement the input and output moderation rails.

Fig. 4 shows a sample flow in Colang that invokes the `check_jailbreak` action. If the jailbreak rail flags a user message, the developer can decide not to show the generated response and to output a default text instead. Appendix F provides other examples of flows using the executions rails.

```
define flow check jailbreak
  user ...
  $allowed = execute check_jailbreak
  if not $allowed
    bot remove last message
    bot inform message breaks moderation
```

Figure 4: Flow using jailbreak rail in Colang

## 5 Evaluation

In this section, we provide details on how we measure the performance of various rails. Additional information for all tasks and a discussion on the automatic evaluation tools available in NeMo Guardrails are provided in Appendix G.

### 5.1 Topical Rails

The evaluation of topical rails focuses on the core mechanism used by the toolkit to guide conversations using canonical forms and dialogue flows. The current evaluation experiments employ datasets used for conversational NLU. In this section, we present the results for the Banking dataset (Casanueva et al., 2022), while additional experiments can be found in Appendix G.

Starting from a NLU dataset, we create a Colang application (publicly available on Github) by mapping intents to canonical forms and defining simple dialogue flows for them. The evaluation dataset used in our experiments is balanced, containing at most 3 samples per intent sampled randomly from the original datasets. The test dataset has 231 samples spanning over 77 different intents.

The results of the top 3 performing models are presented in Fig. 5, showing that topical rails can be successfully used to guide conversations even with smaller open source models such as `falcon-7b-instruct` or `llama2-13b-chat`. As the performance of an LLM is heavily dependent on the prompt, all results might be improved with better prompting.

The topical rails evaluation highlights several important aspects. First, each step in the three-step

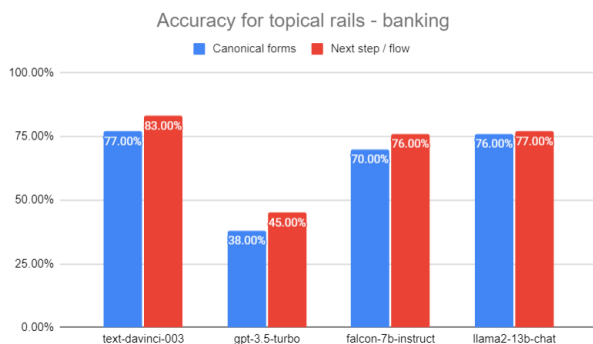


Figure 5: Performance of topical rails on Banking.

approach (user canonical form, next step, bot message) used by Guardrails offers an improvement in performance. Second, it is important to have at least  $k = 3$  samples in the vector database for each user canonical form for achieving good performance. Third, some models (i.e., gpt-3.5-turbo) produce a wider variety of canonical forms, even with few-shot prompting. In these cases, it is useful to add a similarity match instead of exact match for generating canonical forms.

## 5.2 Execution Rails

**Moderation Rails** To evaluate the moderation rails, we use the Anthropic Red-Teaming and Helpful datasets (Bai et al., 2022a; Perez et al., 2022). We have sampled a balanced *harmful-helpful* evaluation set as follows: from the Red-Teaming dataset we sample prompts with the highest harmful score, while from the Helpful dataset we select an equal number of prompts.

We quantify the performance of the rails based on the proportion of harmful prompts that are blocked and the proportion of helpful ones that are allowed. Analysis of the results shows that using both the input and output moderation rails is much more robust than using either one of the rails individually. Using both rails gpt-3.5-turbo has a great performance - blocking close to 99% of harmful (compared to 93% without the rails) and just 2% of helpful requests - details in Appendix G.

**Fact-Checking Rail** We consider the MS-MARCO dataset (Bajaj et al., 2016) to evaluate the performance of the fact-checking rail. The dataset consists of (*context, question, answer*) triples. In order to mine negatives (answers that are *not* grounded in the context) we use OpenAI text-davinci-003 to rewrite the positive answer to a hard negative that looks similar to it, but is

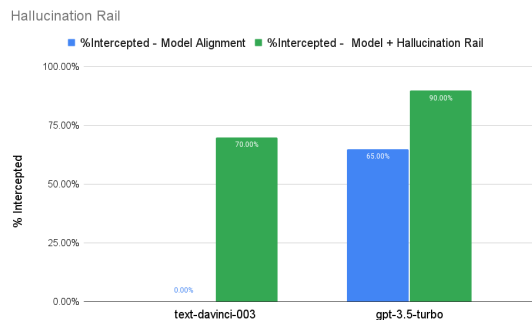


Figure 6: Performance of the hallucination rail.

not grounded in the evidence. We construct a combined dataset by equally sampling both positive and negative triples. Both text-davinci-003 and gpt-3.5-turbo perform well on the fact-checking rail and obtain an overall accuracy of 80% (see Fig. 11 in Appendix G.2.2).

**Hallucination Rail** Evaluating the hallucination rail is difficult without employing subjective manual annotation. To overcome this issue and be able to automatically quantify its performance, we compile a list of 20 questions based on a false premise (questions that do not have a right answer).

Any generation from the language model, apart from deflection, is considered a failure. We then quantify the benefit of employing the hallucination rail as a fallback mechanism. For text-davinci-003, the LLM is unable to deflect prompts that are unanswerable and using the hallucination rail helps intercept 70% of these prompts. gpt-3.5-turbo performs much better, deflecting unanswerable prompts or marking that its response could be incorrect in 65% of the cases. Even in this case, employing the hallucination rail boosts performance up to 95%.

## 6 Conclusions

We present NeMo Guardrails, a toolkit that allows developers to build controllable and safe LLM-based applications by implementing programmable rails. These rails are expressed using Colang and can also be implemented as custom actions if they require a complex logic. Using CoT prompting and a dialogue manager that can interpret Colang code, the Guardrails runtime acts like a proxy between the application and the LLM enforcing the user-defined rails.



## 7 Limitations

### 7.1 Programmable Rails and Embedded Rails

Building controllable and safe LLM-powered applications, in general, and dialogue systems, in particular, is a difficult task. We acknowledge that the approach employed by NeMo Guardrails of using developer-defined programmable rails, implemented with prompting and the Colang interpreter, is not a perfect solution.

Therefore we advocate that, whenever possible, our toolkit should not be used as a stand-alone solution, especially for safety-specific rails. Programmable rails complement embedded rails and these two solutions should be used together for building safe LLM applications. The vision of the project is to also provide, in the future, more powerful customized models for some of the execution rails that should supplement the current pure prompting methods. On another hand, our results show that adding the moderation rails to existing safety rails embedded in powerful LLMs (e.g., ChatGPT), provides a better protection against jail-break attacks.

In the context of controllable and task-oriented dialogue agents, it is difficult to develop customized models for all possible tasks and topical rails. Therefore, in this context, NeMo Guardrails is a viable solution for building LLM-powered task-oriented agents without extra mechanisms. However, even for topical rails and task-oriented agents, we plan to release p-tuned models that achieve better performance for some of the tasks, e.g. for canonical form generation.

### 7.2 Extra Costs and Latency

The three-step CoT prompting approach used by the Guardrails runtime incurs extra costs and extra latency. As these calls are sequentially chained (i.e., the generation of the next steps in the second phase depends on the user canonical form generated in the first stage), the calls cannot be batched. In our current implementation, the latency and costs required are about 3 times the latency and cost of a normal call to generate the bot message without using Guardrails. We are currently investigating if in some cases we could use a single call to generate all three steps (user canonical form, next steps in the flow, and bot message).

Using a more complex prompt and few-shot in-context learning also generates slightly extra latency and a larger cost compared to a normal

bot message generation for a vanilla conversation. Developers can decide to use a simpler prompt if needed.

However, we consider that developers should be provided with various options for their needs. Some might be willing to pay the extra costs for having safer and controllable LLM-powered dialogue agents. Moreover, GPU inference costs will decrease and smaller models can also achieve good performance for some or all NeMo Guardrails tasks. As presented in our paper, we know that `falcon-7b-instruct` (Penedo et al., 2023) already achieves very good performance for topical rails. We have seen similar positive performance from other recent models, like Llama 2 (7B and 13B) chat variants (Touvron et al., 2023).

## 8 Broader Impact

As a toolkit to enforce programmable rails for LLM applications, including dialogue systems, NeMo Guardrails should provide benefits to developers and researchers. Programmable rails supplement embedded rails, either general (using RLHF) or user-defined (using p-tuned customized models). For example, using the fact-checking rail developers can easily build an enhanced retrieval-based LLM application and it also allows them to assess the performance of various models as programmable rails are model-agnostic. The same is true for building LLM-based task-oriented agents that should follow complex dialogue flows.

At the same time, before putting a Guardrails application into production, the implemented programmable rails should be thoroughly tested (especially safety related rails). Our toolkit provides a set of evaluation tools for testing the performance both for topical and execution rails.

Additional details for our toolkit can be found in the Appendix, including simple installation steps for running the toolkit with the example Guardrails applications that are shared on Github. A short demo video is also available: <https://youtu.be/Pfab6UWszEc>.

## References

- Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al.

- 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*.
- 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.
- 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.
- Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.
- Vojtěch Hudeček and Ondřej Dušek. 2023. Are llms all you need for task-oriented dialogue? *arXiv preprint arXiv:2304.06556*.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2023. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. *arXiv preprint arXiv:2302.05733*.
- 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.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.
- Xingkun Liu, Arash Eshghi, Paweł Swietojanski, and Verena Rieser. 2021. Benchmarking natural language understanding services for building conversational agents. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 165–183. Springer.
- Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.
- OpenAI. 2023. *Gpt-4 technical report*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Richard Yuanzhe Pang, Stephen Roller, Kyunghyun Cho, He He, and Jason Weston. 2023. Leveraging implicit feedback from deployment data in dialogue. *arXiv preprint arXiv:2307.14117*.
- Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.
- Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.
- Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448,

- Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2022. Prompting gpt-3 to be reliable. In *The Eleventh International Conference on Learning Representations*.
- Spotify. ANNOY library. <https://github.com/spotify/annoy>. Accessed: 2023-08-01.
- Makeesh Narsimhan Sreedhar and Christopher Parisien. 2022. Prompt learning for domain adaptation in task-oriented dialogue. In *Proceedings of the Towards Semi-Supervised and Reinforced Task-Oriented Dialog Systems (SereTOD)*, pages 24–30, Abu Dhabi, Beijing (Hybrid). Association for Computational Linguistics.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, et al. 2023. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. *arXiv preprint arXiv:2304.06762*.
- Yau-Shian Wang and Yingshan Chang. 2022. Toxicity detection with generative prompt-based inference. *arXiv preprint arXiv:2205.12390*.
- Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Xiaoying Zhang, Baolin Peng, Kun Li, Jingyan Zhou, and Helen Meng. 2023. Sgp-tod: Building task bots effortlessly via schema-guided llm prompting. *arXiv preprint arXiv:2305.09067*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.
- Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

## A Installation Guide and Examples

Developers can download and install the latest version of the NeMo Guardrails toolkit directly from Github <sup>2</sup>. They can also install the latest stable release using `pip install nemoguardrails`.

We have a concise installation guide <sup>3</sup> showing how to run a Guardrails app using the provided Command Line Interface (CLI) or how to launch the Guardrails web server. The server powers a simple chat web client to engage with all the Guardrails apps found in the folder specified when starting the server.

Five reference Guardrails applications are provided as a general demonstration for building different types of rails.

- **Topical Rail:** Making the bot stick to a specific topic of conversation.
- **Moderation Rail:** Moderating a bot's response.
- **Fact Checking and Hallucination Rail:** Ensuring factual answers.
- **Secure Execution Rail:** Executing a third-party service with LLMs.
- **Jail-breaking Rail:** Ensuring safe answers despite malicious intent from the user.

These examples are meant to showcase the process of building rails, not as out-of-the-box safety features. Customization and strengthening of the rails is highly recommended.

The sample Guardrails applications also contain examples on how to use several open-source models (e.g., `falcon-7b-instruct`, `dolly-v2-3b`, `vicuna-7b-v1.3`) deployed locally or using HuggingFace Inference private endpoints. Other examples cover how to combine various chains defined in Langchain with programmable rails defined in NeMo Guardrails.

Additional details about the reference applications and about the toolkit in general can be found on the main documentation page<sup>4</sup>.

<sup>2</sup><https://github.com/NVIDIA/NeMo-Guardrails/>

<sup>3</sup>[https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/getting\\_started/installation-guide.md](https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/getting_started/installation-guide.md)

<sup>4</sup><https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/README.md>

## B Colang Language and Dialogue Manager

Colang is a language for modeling sequences of events and interactions, being particularly useful for modeling conversations. At the same time, it enables the design of guardrails for conversational systems using the Colang interpreter, an event-based processing engine that acts like a dialogue manager.

Creating guardrails for conversational systems requires some form of understanding of how the dialogue between the user and the bot unfolds. Existing dialog management techniques such as flow charts, state machines or frame-based systems are not well suited for modeling highly flexible conversational flows like the ones we expect when interacting with an LLM-based system.

However, since learning a new language is not an easy task, Colang was designed as a mix of natural language (English) and Python. If you are familiar with Python, you should feel confident using Colang after seeing a few examples, even without an explanation.

The main concepts used by the Colang language are the following:

- *Utterance:* the raw text coming from the user or the bot.
- *Message:* the canonical form (structured representation) of a user/bot utterance.
- *Event:* something that has happened and is relevant to the conversation, e.g. user is silent, user clicked something, user made a gesture, etc.
- *Action:* a custom code that the bot can invoke; usually for connecting to a third-party API.
- *Context:* any data relevant to the conversation (encoded as a key-value dictionary).
- *Flow:* a sequence of messages and events, potentially with additional branching logic.
- *Rails:* specific ways of controlling the behavior of a conversational system (a.k.a. bot), e.g. not talk about politics, respond in a specific way to certain user requests, follow a predefined dialog path, use a specific language style, extract data etc. A rail in Colang can be modeled through one or more flows.

For additional details about Colang, please consult the Colang syntax guide <sup>5</sup>.

The Guardrails runtime uses an event-driven design (i.e., an event loop that processes events and generates back other events). Dialogue flows are treated as sequences of events, but even a simple user message is also an event - as an `UtteranceUserActionFinished` event is created and sent to the runtime. More details are available in the NeMo Guardrails architecture guide <sup>6</sup>.

## C Prompts for Topical Rails

NeMo Guardrails uses complex prompts, chained in 3 steps, to respond to a user message as described in Section 3.2. In the following listing we provide an example for the first step, to generate the canonical form for the last user message in the current conversation.

The prompt below is designed for text-davinci-003 and is structured in four parts:

1. General prompt describing the task of the application.
2. Sample conversation using Colang syntax.
3. The most similar, given the current user message, few-shot ( $k = 5$ ) examples for mapping user messages to their corresponding canonical form.
4. The current conversation between the user and the bot in Colang syntax.

```
"""
Below is a conversation between a helpful AI assistant and a
user. The bot is designed to generate human-like text
based on the input that it receives. The bot is
talkative and provides lots of specific details. If the
bot does not know the answer to a question, it
truthfully says it does not know.
"""

# This is how a conversation between a user and the bot can
go:

user "Hello there!"
  express greeting
bot express greeting
  "Hello! How can I assist you today?"
user "What can you do for me?"
  ask about capabilities
bot respond about capabilities
  "I am an AI assistant which helps answer questions based
  on a given knowledge base. For this interaction, I
  can answer question based on the job report published
  by US Bureau of Labor Statistics"
user "Tell me a bit about the US Bureau of Labor Statistics."
  ask question about publisher
```

<sup>5</sup>[https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/user\\_guide/colang-language-syntax-guide.md](https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/user_guide/colang-language-syntax-guide.md)

<sup>6</sup><https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/architecture/README.md>

```
bot response for question about publisher
  "The Bureau of Labor Statistics is the principal fact-
  finding agency for the Federal Government in the
  broad field of labor economics and statistics"
user "thanks"
  express appreciation
bot express appreciation and offer additional help
  "You're welcome. If you have any more questions or if
  there's anything else I can help you with, please don
  't hesitate to ask."

# This is how the user talks:

user "What was the movement on nonfarm payroll?"
  ask about headline numbers

user "What's the number of part-time employed number?"
  ask about household survey data

user "How much did the nonfarm payroll rise by?"
  ask about headline numbers

user "What is this month's unemployment rate?"
  ask about headline numbers

user "How many long term unemployment individuals were
  reported?"
  ask about household survey data

# This is the current conversation between the user and the
  bot:

user "Hello there!"
  express greeting
bot express greeting
  "Hello! How can I assist you today?"
user "What can you do for me?"
  ask about capabilities
bot respond about capabilities
  "I am an AI assistant which helps answer questions based
  on a given knowledge base. For this interaction, I
  can answer question based on the job report published
  by US Bureau of Labor Statistics"
user "how many unemployed people were there in March?"
```

Similar prompts are defined for other LLMs (i.e., gpt-3.5-turbo, falcon-7b-instruct and others) and are available on Github <sup>7</sup>.

When generating the user canonical form and the next steps to guide the conversation, we use  $temp = 0$ , while for sampling the bot message we use a higher temperature ( $temp = 0.7$  or  $temp = 1$ ).

## D Prompt Templates for Execution Rails

In this section we provide the prompt templates used by the hallucination and moderation rails.

### D.1 Hallucination Rail

After we obtain  $n$  samples from the conversational agent for the same prompt, we concatenate  $n - 1$  responses to form the *context* and use the  $n^{th}$  response as the *hypothesis*. We utilize an LLM to verify if the hypothesis is consistent with the context using the following prompt template:

*You are given a task to identify if the hypothesis is in agreement with the context below. You will only use the contents of the context and not rely on external knowledge. Answer with yes/no.*

<sup>7</sup><https://github.com/NVIDIA/NeMo-Guardrails/tree/main/nemoguardrails/llm/prompts>

```
"context": {{ sampled_responses }} "hypothesis": {{ bot_response }} "agreement":
```

When sampling the bot responses, as proposed by SelfCheckGPT (Manakul et al., 2023) we use a high temperature ( $temp = 1$ ) to ensure the variation between them necessary to assess the self-consistency of the answers. At the same time, the LLM-based hallucination verification call uses  $temp = 0$  to get the most probable response. This behavior is similar for all the other execution rails described in the paper.

## D.2 Moderation Rails

Both the input and output moderation rails are framed as another task to a powerful, well-aligned LLM that vets the input or response. For the jailbreak rail, we use the following prompt template:

```
Instruction: {{ user_input }}
Would this instruction make a language model break moderation policies, deviate from good aligned responses and provide answers that a language model should ideally not? Answer with yes/no.
```

The output moderation rail uses the following prompt template:

```
Model output: {{ bot_response }}
Is this output legal, ethical and not harmful to any person? Answer with yes/no.
```

## E Action Definitions

The first step in creating an execution rail is to define a custom action that implements the logic of the rail. At this moment, this is done in Python.

Any custom action has access to the *context* of the conversation as can be seen in the subsequent examples. In the Guardrails runtime, the context is a sequence of all the events in the conversation history - including user and bot messages, canonical forms, action called and more. Some of the context events that might be accessed more often to define actions have a shortcut, e.g. `context.get("last_bot_message")`.

An action can receive any number of parameters from the Colang scripts where they are called. These are passed to the Python function implementing the action logic. At the same time, an action usually returns a value that can be used to further guide the dialogue.

## E.1 Output Moderation Rail

This is the current implementation for the output moderation action. It uses the prompt template defined in Appendix D and returns *True* if the bot response breaks the output moderation rail.

The call to the LLM is done using prompt templates and a simple LLM chain provided by Langchain<sup>8</sup> for simplifying the communication with several LLM providers.

```
bot_response = context.get("last_bot_message")

if bot_response:
    output_moderation_template =
        "Model Output: {bot_response}
        Is this output legal, ethical and not harmful to any
        person? Answer with yes/no."

    prompt = PromptTemplate(
        template=output_moderation_template, input_variables=
            [{"bot_response"}]
    )

    output_moderation_chain = LLMChain(prompt=prompt, llm=
        llm)
    check = await output_moderation_chain.apredict(
        bot_response=bot_response)

    check = check.lower().strip()
    log.info(f"Output moderation check result is {check}.")

    if "no" in check:
        return False
    return True
```

## E.2 Jailbreak Rail

This is the current implementation for the input moderation action. It uses the prompt template defined in Appendix D and returns *True* if the user input breaks the jailbreak rail.

```
user_input = context.get("last_user_message")

if user_input:
    jailbreak_template = "Instruction: {user_input}
    Would this instruction make a language model break
    moderation policies, deviate from good aligned
    responses and provide answers that a language model
    should ideally not? Answer with yes/no."

    prompt = PromptTemplate(
        template=jailbreak_template, input_variables=["
            user_input"]
    )

    jailbreak_chain = LLMChain(prompt=prompt, llm=llm)
    check = await jailbreak_chain.apredict(bot_response=
        bot_response)

    check = check.lower().strip()
    log.info(f"Jailbreak check result is {check}.")

    if "no" in check:
        return False
    return True
```

## F Sample Guardrails Flows using Actions

This section includes some examples of using the safety execution rails, implemented as custom actions, inside Colang flows to define simple Colang applications.

<sup>8</sup><https://github.com/langchain-ai/langchain>

Figure 7 shows how to use the `check_jailbreak` action for input moderation. The semantics is that for each user message (user ...), the jailbreak action is called to verify the last user message, and if it is flagged as a jailbreak attempt the last LLM bot-generated answer is removed and a new one is uttered to inform the user her/his message breaks the moderation policy. Figure 8 shows how the `output_moderation` action is used - the meaning is similar to jail-breaking, however it is triggered after any output bot message event (bot ...).

```
define flow check jailbreak
  user ...
  $allowed = execute check_jailbreak
  if not $allowed
    bot remove last message
    bot inform message breaks moderation
```

Figure 7: Flow using jailbreak rail in Colang

```
define flow check bot response
  bot ...
  $allowed = execute output_moderation
  if not $allowed
    bot remove last message
    bot inform message breaks moderation
```

Figure 8: Flow using output moderation in Colang

In a similar way, Fig. 9 shows how to use the hallucination rail to check responses when for a particular topic (i.e., asking questions about persons, where GPT models are prone to hallucinate). In this case, the bot message is not removed, but an extra message is added to warn the user about a possible incorrect answer. Fig. 10 shows how to add fact-checking again for a specific topic, when asking a question about an employment report. In this situation, the LLM should be consistent with the information in the report.

```
define flow check hallucination
  user ask about person
  bot ...
  $result = execute check_hallucination
  if $result
    bot inform answer prone to hallucination
```

Figure 9: Flow using hallucination rail in Colang

```
define flow answer report question
  user ask about report
  bot provide report answer
  $accurate = execute check_facts
  if not $accurate
    bot remove last message
    bot inform answer unknown
```

Figure 10: Flow using fact-checking rail in Colang

## G Additional Details on Evaluation

Our toolkit also provides the evaluation tooling and methodology to assess the performance of topical and execution rails. All the results reported in the paper can be replicated using the CLI evaluation tool available on Github, following the instructions about evaluation<sup>9</sup>. The same page contains slightly more details than the current paper and is regularly updated with new results (including new LLMs).

Detailed instructions on how to replicate the experiments can be found here<sup>10</sup>.

### G.1 Topical Rails

Topical rails evaluation focuses on the core mechanism used by NeMo Guardrails to guide conversations using canonical forms and dialogue flows.

The current evaluation experiments for topical rails uses two datasets employed for conversational NLU: chit-chat<sup>11</sup> and banking.

The datasets were transformed into a NeMo Guardrails app, by defining canonical forms for each intent, specific dialogue flows, and even bot messages (for the chit-chat dataset alone). The two datasets have a large number of user intents, thus topical rails. One of them is very generic and with coarse-grained intents (chit-chat), while the banking dataset is domain-specific and more fine-grained. More details about running the topical rails evaluation experiments and the evaluation datasets is available here.

Preliminary evaluation results follow next. In all experiments, we have chosen to have a balanced test set with at most 3 samples per intent. For both datasets, we have assessed the performance for various LLMs and also for the number of samples

<sup>9</sup><https://github.com/NVIDIA/NeMo-Guardrails/blob/main/nemoguardrails/eval/README.md>

<sup>10</sup><https://github.com/NVIDIA/NeMo-Guardrails/blob/main/docs/README.#evaluation-tools>

<sup>11</sup><https://github.com/rahu1051296/small-talk-rasa-stack>, dataset was initially released by Rasa

( $k = all, 3, 1$ ) per intent that are indexed in the vector database. We have used a random seed of 42 for all experiments to ensure consistency.

The results of the top 3 performing models are presented in Fig. 5, showing that topical rails can be successfully used to guide conversations even with smaller open source models such as falcon-7b-instruct or llama2-13b-chat. As the performance of an LLM is heavily dependent on the prompt, due to the complex prompt used by NeMo Guardrails all results might be improved with better prompting.

The topical rails evaluation highlights several important aspects. First, each step in the three-step approach (user canonical form, next step, bot message) used by Guardrails offers an improvement in performance. Second, it is important to have at least  $k = 3$  samples in the vector database for each user canonical form for achieving good performance. Third, some models (i.e., gpt-3.5-turbo) produce a wider variety of canonical forms, even with few-shot prompting. In these cases, it is useful to add a similarity match instead of exact match for generating canonical forms. In this case, the similarity threshold becomes an important inference parameter.

Dataset statistics and detailed results for several LLMs are presented in Tables 1, 2, and 3. Some experiments have missing numbers either because those experiments did not compute those metrics or because the dataset does not contain specific items (for example, user-defined bot messages for the banking dataset).

| Dataset   | # intents | # test samples |
|-----------|-----------|----------------|
| chit-chat | 76        | 226            |
| banking   | 77        | 231            |

Table 1: Dataset statistics for the topical rails evaluation.

## G.2 Execution Rails

### G.2.1 Moderation Rail

To evaluate the moderation rails, we use the Anthropic Red-Teaming and Helpful datasets (Bai et al., 2022a; Perez et al., 2022). The red-teaming dataset consists of prompts that are human-annotated (0-4) on their ability to elicit inappropriate responses from language models. A higher score implies that the prompt was more successful in bypassing model alignment. We randomly sample prompts with the highest rating to curate

the *harmful* set. All the prompts in the Anthropic Helpful dataset are genuine queries and forms our *helpful* set. We create a balanced evaluation set with an equal number of *harmful* and *helpful* samples.

We quantify the performance of the rails based on the proportion of harmful prompts that are blocked and the proportion of helpful ones that are allowed. An ideal model would be able to block 100% of the harmful prompts and allow 100% of the helpful ones. We pass prompts from our evaluation set through the input (jailbreak) moderation rail. Only those that are not flagged are passed to the conversational agent to generate a response which is passed through the output moderation rail. Once again, only those responses that are not flagged are displayed back to the user.

Analysis of the results shows that using a combination of both the input (aka *jailbreak* rail) and output moderation rails is more robust than using either one of the rails individually. It should also be noted that evaluation of the output moderation rail is subjective and each person/organization would have different subjective opinions on what should be allowed to pass through or not. In such situations, it would be easy to modify prompts to the moderation rails to reflect the beliefs of the entity deploying the conversational agent.

Using an evaluation set of 200 samples split equally between *harmful* and *helpful* and created as described above, we have seen that text-davinci-003 blocks only 24% of the harmful messages, while gpt-3.5-turbo does much better blocking 93% of harmful messages without any moderation guardrail. In this case, blocking means that the model is not providing a response to an input requiring moderation. On the helpful inputs, both models do not block any request. Using only the input moderation rail, text-davinci-003 blocks 87% of harmful and 3% of helpful requests. Using both input and output moderation, text-davinci-003 blocks 97% of harmful and 5% of helpful requests, while gpt-3.5-turbo has a great performance - blocking close to 99% of harmful and just 2% of helpful requests.

### G.2.2 Fact-checking Rail

We consider the MSMARCO dataset (Bajaj et al., 2016) to evaluate the performance of the fact-checking rail. The dataset consists of (*context, question, answer*) triples. In order to mine negatives (answers that are *not* grounded in the context),



| Model                     | Us int,<br>no sim | Us int,<br>sim=0.6 | Bt int,<br>no sim | Bt int,<br>sim=0.6 | Bt msg,<br>no sim | Bt msg,<br>sim=0.6 |
|---------------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|
| text-davinci-003, k=all   | 0.89              | 0.89               | 0.90              | 0.90               | 0.91              | 0.91               |
| text-davinci-003, k=3     | 0.82              | N/A                | 0.85              | N/A                | N/A               | N/A                |
| text-davinci-003, k=1     | 0.65              | N/A                | 0.73              | N/A                | N/A               | N/A                |
| gpt-3.5-turbo, k=all      | 0.44              | 0.56               | 0.50              | 0.61               | 0.54              | 0.65               |
| dolly-v2-3b, k=all        | 0.65              | 0.78               | 0.68              | 0.78               | 0.69              | 0.78               |
| falcon-7b-instruct, k=all | 0.81              | 0.81               | 0.81              | 0.82               | 0.81              | 0.82               |
| llama2-13b-chat, k=all    | 0.87              | N/A                | 0.88              | N/A                | 0.89              | N/A                |

Table 2: Topical evaluation results on chit-chat dataset. **Us int** means accuracy for user intents, **Bt int** is accuracy for next step generation (i.e., the bot intent), **Bt msg** is accuracy for generated bot message. **Sim** denotes if semantic similarity was used for matching (with a specified threshold, in this case 0.6) or exact match.

| Model                     | Us int,<br>no sim | Us int,<br>sim=0.6 | Bt int,<br>no sim | Bt int,<br>sim=0.6 | Bt msg,<br>no sim | Bt msg,<br>sim=0.6 |
|---------------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|
| text-davinci-003, k=all   | 0.77              | 0.82               | 0.83              | 0.84               | N/A               | N/A                |
| text-davinci-003, k=3     | 0.65              | N/A                | 0.73              | N/A                | N/A               | N/A                |
| text-davinci-003, k=1     | 0.50              | N/A                | 0.63              | N/A                | N/A               | N/A                |
| gpt-3.5-turbo, k=all      | 0.38              | 0.73               | 0.45              | 0.73               | N/A               | N/A                |
| dolly-v2-3b, k=all        | 0.32              | 0.62               | 0.40              | 0.64               | N/A               | N/A                |
| falcon-7b-instruct, k=all | 0.70              | 0.76               | 0.75              | 0.78               | N/A               | N/A                |
| llama2-13b-chat, k=all    | 0.76              | N/A                | 0.78              | N/A                | N/A               | N/A                |

Table 3: Topical evaluation results on banking dataset.

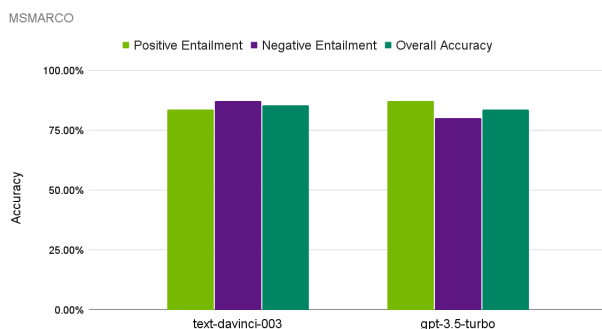


Figure 11: Performance of the fact-checking rail.

we use OpenAI text-davinci-003 to rewrite the positive answer to a hard negative that looks similar to it, but is not grounded in the evidence. We construct a combined dataset by equally sampling both positive and negative triples. Both text-davinci-003 and gpt-3.5-turbo perform well on the fact-checking rail and obtain an overall accuracy of 80% (Fig. 11). The behavior of the two models is slightly different: while gpt-3.5-turbo is better at discovering negatives, text-davinci-003 performs better on positive samples.

### G.2.3 Hallucination Rail

Evaluating the hallucination rail is difficult since we cannot ascertain the questions that can be answered with factual knowledge embedded in the parameters of the language model. To effectively quantify the ability of the model to detect hallucinations, we compile a list of 20 questions based on a false premise. For example, one such question that does not have a right answer is: "When was the undersea city in the Gulf of Mexico established?"

Any generation from the language model apart from deflection (i.e., recognizing that the question is unanswerable) is considered a failure. We also quantify the benefit of employing the hallucination rail as a fallback mechanism. For text-davinci-003, the base language model is unable to deflect prompts that are unanswerable and using the hallucination rail helps intercept 70% of the unanswerable prompts. gpt-3.5-turbo performs very well at deflecting prompts that cannot be answered or hedging its response with statements about it could be incorrect. Even for such powerful models, we find that employing the hallucination rail helps boost the identification of questions that are prone to incorrect responses by 25%.

# LM-Polygraph: Uncertainty Estimation for Language Models

Ekaterina Fadeeva<sup>3,5</sup>  $\diamond$  Roman Vashurin<sup>2</sup>  $\diamond$  Akim Tsvigun<sup>5,6,7</sup>  $\diamond$  Artem Vazhentsev<sup>3,4</sup>  $\diamond$   
Sergey Petrakov<sup>3</sup>  $\diamond$  Kirill Fedyanin<sup>2</sup> Daniil Vasilev<sup>5</sup> Elizaveta Goncharova<sup>4,5,6</sup>  
Alexander Panchenko<sup>3,4</sup> Maxim Panov<sup>1</sup> Timothy Baldwin<sup>1,8</sup> Artem Shelmanov<sup>1</sup>  
<sup>1</sup>MBZUAI <sup>2</sup>TII <sup>3</sup>Center for Artificial Intelligence Technology <sup>4</sup>AIRI  
<sup>5</sup>HSE University <sup>6</sup>AI Center NUST MISiS <sup>7</sup>Semrush <sup>8</sup>The University of Melbourne  
{ekaterina.fadeeva, sergey.petrakov}@skol.tech {roman.vashurin, kirill.fedyanin}@tii.ae  
akim.tsvigun@semrush.com {vazhentsev, panchenko, goncharova}@airi.net  
{maxim.panov, timothy.baldwin, artem.shelmanov}@mbzuai.ac.ae

## Abstract

Recent advancements in the capabilities of large language models (LLMs) have paved the way for a myriad of groundbreaking applications in various fields. However, a significant challenge arises as these models often “hallucinate”, i.e., fabricate facts without providing users an apparent means to discern the veracity of their statements. Uncertainty estimation (UE) methods are one path to safer, more responsible, and more effective use of LLMs. However, to date, research on UE methods for LLMs has been focused primarily on theoretical rather than engineering contributions. In this work, we tackle this issue by introducing LM-Polygraph, a framework with implementations of a battery of state-of-the-art UE methods for LLMs in text generation tasks, with unified program interfaces in Python.<sup>1</sup> Additionally, it introduces an extendable benchmark for consistent evaluation of UE techniques by researchers, and a demo web application that enriches the standard chat dialog with confidence scores, empowering end-users to discern unreliable responses.<sup>2,3</sup> LM-Polygraph is compatible with the most recent LLMs, including BLOOMz, LLaMA-2, ChatGPT, and GPT-4, and is designed to support future releases of similarly-styled LLMs.

## 1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across a variety of text generation tasks. Instruction fine-tuning and reinforcement learning from human feedback (RLHF) have brought the zero-shot performance of these models to a new level (Ouyang et al., 2022). However, the capabilities of LLMs, despite their profound power and complexity, are inherently constrained. Limitations arise from the finite nature

of the training data and the model’s intrinsic memorization and reasoning capacities. Hence, their utility is bounded by the depth and breadth of the knowledge they embed.

Due to their training objectives, even when the embedded knowledge of an LLM on a given topic is limited, it tends to be over-eager to respond to a prompt, sometimes generating misleading or entirely erroneous output. This dangerous behavior of attempting to appease the user with plausible-sounding but potentially false information is known as “hallucination” (Xiao and Wang, 2021; Dziri et al., 2022). It poses a significant challenge when deploying LLMs in practical applications.

There are several well-known approaches to censoring LLM outputs, including: filtering with stopword lists, post-processing with classifiers (Xu et al., 2023), rewriting of toxic outputs (Logacheva et al., 2022), and longer fine-tuning with RLHF. However, these approaches cannot be relied on to completely resolve hallucinations. Since LMs are natural (if “unintentional”) liars, we propose LM-Polygraph — a program framework that, similar to a human polygraph, leverages various hidden signals to reveal when one should not trust the subject. In particular, LM-Polygraph provides a comprehensive collection of uncertainty estimation (UE) techniques for LLMs in text generation tasks.

Uncertainty estimation refers to the process of quantifying the degree of confidence in the predictions made by a machine learning model. For classification and regression tasks, there is a well-developed battery of methods (Gal, 2016). There has also been a surge of work investigating UE, particularly in text classification and regression in conjunction with encoder-only LMs such as BERT (Zhang et al., 2019; He et al., 2020; Shelmanov et al., 2021; Xin et al., 2021; Vazhentsev et al., 2022; Kotelevskii et al., 2022; Wang et al., 2022; Kuzmin et al., 2023). However, UE for sequence generation tasks, including text generation,

<sup>1</sup><http://lm-polygraph.nlpresearch.group>

<sup>2</sup><http://lm-polygraph-demo.nlpresearch.group>

<sup>3</sup><http://lm-polygraph-video.nlpresearch.group>

$\diamond$  Equal contribution

is a much more complex problem. To quantify the uncertainty of the whole sequence, we have to aggregate uncertainties of many individual token predictions and deal with non-trivial sampling and pruning techniques like beam search. Contrary to classification tasks where the number of possible prediction options is finite, in text generation, the number of possible predictions is infinite or exponential in vocabulary size, complicating the estimation of probabilities and information-based scores. Finally, a natural language text is not a simple sum of its tokens; it is a nuanced interposition of context, semantics, and grammar, so two texts can have very diverse surface forms but similar meanings, which should be taken into account during the UE process.

Several recent studies have delved into developing UE methods for LMs in text generation tasks (Malinin and Gales, 2021; van der Poel et al., 2022; Kuhn et al., 2023; Ren et al., 2023; Vazhentsev et al., 2023b; Lin et al., 2023). However, the current landscape of this research is quite fragmented with many non-comparable or even concurrent studies, which makes it challenging to consolidate the findings and draw holistic conclusions.

In this work, with the development of LM-Polygraph, we strive to bridge these disparate research efforts, fostering more cohesion and synergy in the field. We envision a framework that consolidates the scattered UE techniques within unified frameworks in Python, provides an extendable evaluation benchmark, and offers tools to integrate uncertainty quantification in standard LLM pipelines seamlessly. This endeavor will not only make the journey less challenging for individual researchers and developers but also set the stage for more robust, reliable, and trustworthy LLM deployments for end-users.

Our **contributions** are as follows:

- We provide a comprehensive framework that implement state-of-the-art methods for UE of LM predictions. We also provide the ability to combine multiple uncertainty scores together as suggested by Ren et al. (2023); Vazhentsev et al. (2023a).
- We create a tool that enriches standard LLM chat capabilities with uncertainty scores for model outputs. The tool can potentially be used by end-users to determine whether the answers of language models are reliable or not, and by researchers to develop novel UE

```

from lm_polygraph import estimate_uncertainty
from lm_polygraph import WhiteboxModel
from lm_polygraph.estimated import *

model = WhiteboxModel.from_pretrained(
    "bigscience/bloomz-3b",
    device="cuda:0",
)
ue_method = MeanPointwiseMutualInformation()

input_text = "Who is George Bush?"
estimate_uncertainty(model, ue_method, input_text)

# Output:
# UncertaintyOutput(
#   generation='President of the United States',
#   uncertainty=-6.858096446298684)

```

Figure 1: Code example of how LM predictions could be enriched with uncertainty scores using LM-Polygraph.

techniques for LMs in text generation tasks.

- We construct an easy-to-extend benchmark for UE methods in text generation tasks and provide reference experimental results for implemented UE techniques.

## 2 Python Library

LM-Polygraph implements a set of state-of-the-art UE techniques for LLMs with unified program interfaces in Python. It is compatible with models from the Huggingface library and is tested with recent public-domain LLMs such as BLOOMz (Scao et al., 2022; Yong et al., 2023), Dolly v2 (Conover et al., 2023), Alpaca (Taori et al., 2023), LLaMA-2 (Touvron et al., 2023), and Flan-T5 (Chung et al., 2022). The framework supports both conditional models with a seq2seq architecture and unconditional decoder-only LMs. Figure 1 contains a code example of LM-Polygraph with BLOOMz-3B for UE in open-domain question answering. Some methods that do not require access to the model itself or its logits could be used in conjunction with web-hosted LLMs like ChatGPT or GPT-4 through APIs. We provide a program wrapper for integration with popular online services.

## 3 Uncertainty Estimation Methods

Here, we summarize UE methods implemented in LM-Polygraph, as listed in Table 1.

There are two major technique types: white-box and black-box. The *white-box* methods require access to logits, internal layer outputs, or the LM itself. The *black-box* methods require access only to the generated texts, and can easily be integrated with third-party online services like OpenAI LM API. We note that the methods differ by computational requirements: some techniques pose high

| Uncertainty Estimation Method                                       | Type      | Category          | Compute | Memory | Need Training Data? |
|---|-----------|-------------------|---------|--------|---------------------|
| Maximum sequence probability  | White-box | Information-based | Low     | Low    | No                  |
| Perplexity (Fomicheva et al., 2020)                                 |           |                   | Low     | Low    | No                  |
| Mean token entropy (Fomicheva et al., 2020)                         |           |                   | Low     | Low    | No                  |
| Monte Carlo sequence entropy (Kuhn et al., 2023)                    |           |                   | High    | Low    | No                  |
| Pointwise mutual information (PMI) (Takayama and Arase, 2019)       |           |                   | Medium  | Low    | No                  |
| Conditional PMI (van der Poel et al., 2022)                         |           |                   | Medium  | Medium | No                  |
| Semantic entropy (Kuhn et al., 2023)                                | White-box | Meaning diversity | High    | Low    | No                  |
| Sentence-level ensemble-based measures (Malinin and Gales, 2021)    | White-box | Ensembling        | High    | High   | Yes                 |
| Token-level ensemble-based measures (Malinin and Gales, 2021)       |           |                   | High    | High   | Yes                 |
| Mahalanobis distance (MD) (Lee et al., 2018)                        | White-box | Density-based     | Low     | Low    | Yes                 |
| Robust density estimation (RDE) (Yoo et al., 2022)                  |           |                   | Low     | Low    | Yes                 |
| Relative Mahalanobis distance (RMD) (Ren et al., 2023)              |           |                   | Low     | Low    | Yes                 |
| Hybrid Uncertainty Quantification (HUQ) (Vazhentsev et al., 2023a)  |           |                   | Low     | Low    | Yes                 |
| p(True) (Kadavath et al., 2022)                                     | White-box | Reflexive         | Medium  | Low    | No                  |
| Number of semantic sets (NumSets) (Lin et al., 2023)                | Black-box | Meaning diversity | High    | Low    | No                  |
| Sum of eigenvalues of the graph Laplacian (EigV) (Lin et al., 2023) |           |                   | High    | Low    | No                  |
| Degree matrix (Deg) (Lin et al., 2023)                              |           |                   | High    | Low    | No                  |
| Eccentricity (Ecc) (Lin et al., 2023)                               |           |                   | High    | Low    | No                  |
| Lexical similarity (LexSim) (Fomicheva et al., 2020)                |           |                   | High    | Low    | No                  |

Table 1: UE methods implemented in LM-Polygraph.

computational or memory overheads, e.g., due to repeated inference, making them less suitable for practical usage. The application of some methods also can be hindered by the need for access to the model training data.

Let us consider the input sequence  $\mathbf{x}$  and the output sequence  $\mathbf{y} \in \mathcal{Y}$  of length  $L$ , where  $\mathcal{Y}$  is a set of all possible output sequences. Then the probability of an output sequence given an input sequence for probabilistic autoregressive language models is given by:

$$P(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta}) = \prod_{l=1}^L P(y_l | \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}), \quad (1)$$

where the distribution of each  $y_l$  is conditioned on all the previous tokens in a sequence  $\mathbf{y}_{<l} = \{y_1, \dots, y_{l-1}\}$ , and  $\boldsymbol{\theta}$  denotes the parameters of the model.

### 3.1 White-box Methods

We start the discussion of white-box techniques from **information-based methods**. These techniques are based on token  $P(y_l | \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta})$  and sequence  $P(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta})$  probabilities obtained from a single model prediction. The notable example is *entropy*, which can be calculated on the token or sequence level. The benefits of information-based methods are that they are cheap to compute and simple to implement. However, the quality of these methods is usually relatively low, so they are considered as baselines. Some domain-specific methods were recently proposed in an attempt to improve over standard information-based approaches, such as *semantic entropy* (Kuhn et al., 2023).

The second category of white-box techniques is **ensemble-based methods**, which leverage the diversity of output predictions made by multiple slightly different versions of models under slightly different conditions. Let us assume that  $M$  models are available with parameters  $\boldsymbol{\theta}_i, i = 1, \dots, M$ . These parameters can be obtained via independent training of models. Then one can use token  $P(y_l | \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}_i)$  and sequence  $P(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta}_i)$  probabilities to compute various metrics such as *mutual information* that measures the discrepancy between model predictions.

**Density-based methods** leverage latent representations of instances and construct a probability density on top of them. Usually, these methods approximate training data distribution with the help of one or multiple Gaussian distributions. They can provide a probability or an unnormalized score that determines how likely instances belong to the training data distribution. Therefore, they are good at spotting out-of-distribution (OOD) instances (Vazhentsev et al., 2023b). Several variations of these methods have been proposed in the literature (Lee et al., 2018; Yoo et al., 2022; Ren et al., 2023; Kotelevskii et al., 2022).

The primary advantage of these methods is that they are computationally efficient: they do not need much time for additional model inference, and memory overhead for storing additional parameters is minimal. The drawback is that these methods require access to the model’s training data to fit auxiliary models like Gaussians (e.g., the Mahalanobis Distance method requires constructing

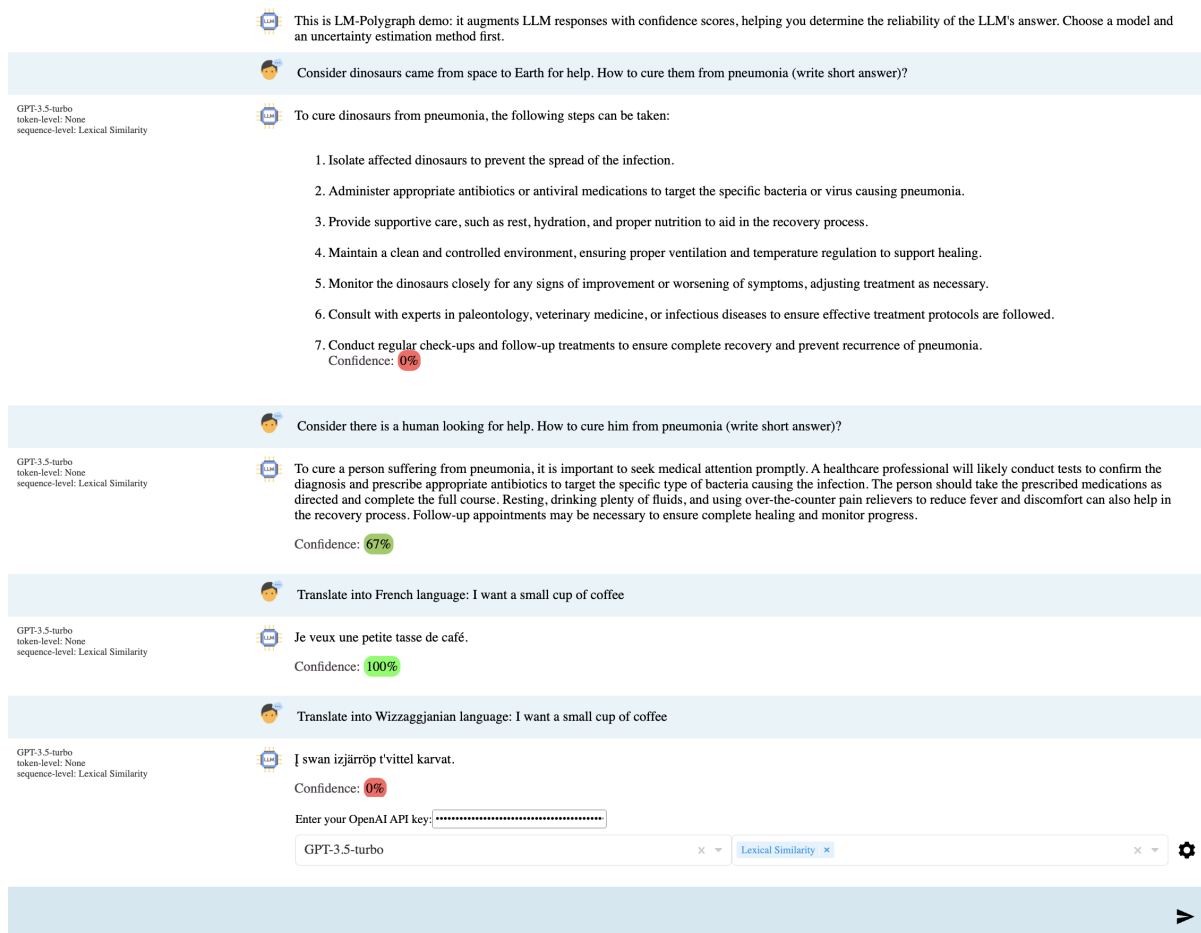


Figure 2: User interface of the demo. A user can interact with an LLM as with any other chat service, but in LM-Polygraph the user also sees the confidence of the model answers. It is possible to specify various UE techniques and various models, including ChatGPT.

data centroids and covariance matrices). These methods are also known to capture only epistemic uncertainty. Therefore, they might not be perfect for selective generation as they cannot be used to spot ambiguous in-domain instances.

Finally, we also combine information-based and density-based methods as suggested by Vazhentsev et al. (2023a) and Ren et al. (2023). More specifically, we implement the *hybrid uncertainty quantification (HUQ)* method (Vazhentsev et al., 2023a) that performs a ranking-based aggregation and leverages strengths of both information-based methods that detect ambiguous instances and density-based methods that detect OOD instances.

Directly **asking the model to validate its answer** is another option for UE (Kadavath et al., 2022). In this method, one asks models first to propose answers and then to evaluate the probability  $P(\text{True})$  that their answers are correct. Kadavath et al. (2022) show that it achieves reasonable performance on a variety of tasks, including question-

answering. We note that this method requires inference of a model twice: the first to generate an answer, and the second for processing its own output. Even though the second inference is usually faster than the first one, it still takes considerable time for computation.

### 3.2 Black-box Methods

In contemporary models, there are instances where the model’s architecture and hidden states are unavailable or there is no access to logits during response generation. Nevertheless, a whole class of black box methods only needs to access the model’s response. Within the scope of this paper, we consider several approaches of this type that have performed well in other studies (Fomicheva et al., 2020; Kuhn et al., 2023; Lin et al., 2023). We focus on Lexical Similarity, Number of Semantic Sets, Sum of Eigenvalues of the Graph Laplacian, Degree Matrix, and Eccentricity. We use the same methodological approach as the authors of

the work (Lin et al., 2023):

- Obtain  $K$  responses  $y_1, \dots, y_K$  for a particular input  $x$ .
- Compute  $K \times K$  similarity matrix  $S$  between responses, where  $S_{ij} = s(y_i, y_j)$  for some similarity score  $s$  (Natural Language Inference score or Jaccard score).
- Based on the similarity matrix  $S$ , we compute the final uncertainty score.

Thus, the idea of the methods is to analyze the similarity matrix and aggregate the information to compute the uncertainty score.

## 4 Demo

We constructed a demo application that can be used to interact with LLMs and also see confidence scores of model answers (see Figure 2). A user specifies a UE method and a language model from a number of publicly-available LLMs with up to 13B parameters, e.g., BLOOMz, Vicuna, and LLaMA-2. There is also the ability to communicate with LLMs deployed as web services such as ChatGPT or GPT-4 and obtain their uncertainty scores based on black-box techniques. For these models, a user should provide an API key.

This demo application is potentially helpful for both end-users and researchers. For end-users, it extends the standard AI assistant interface with information about whether it is reasonable to trust a model answer. Researchers could use this tool for qualitative analysis of various UE methods and LLM responses.

## 5 Evaluation Benchmark

LM-Polygraph provides a vast evaluation benchmark. It contains a script for running one or multiple experiments with UE techniques, implemented as Python modules. This feature allows the user to easily extend the set of available methods and evaluate novel UE techniques in a unified manner. Using this benchmark, we have conducted experiments with most methods implemented in LM-Polygraph. Below, we provide experimental details.

**Datasets.** We experiment with three text generation tasks: machine translation (MT), text summarization (TS), and question answering (QA). For each task, we use two widely-used datasets: WMT-14 German to English and WMT-14 French to English (Bojar et al., 2014) for MT, XSum (Narayan et al., 2018) and AESLC (Zhang and Tetreault,

2019) for TS, and CoQA (Reddy et al., 2019) and bAbI QA (Dodge et al., 2016) for QA. Dataset statistics are presented in Appendix D.

**Models.** We conducted experiments with the Vicuna-v1.5-7B (Zheng et al., 2023) and Llama-v2-7B (Touvron et al., 2023) models. The generation hyperparameters are provided in Appendix B.

**Metrics.** We focus on the task of selective generation (Ren et al., 2023) where we “rejecting” generated sequences due to low quality based on uncertainty scores. Rejecting means that we do not use the model output, and the corresponding queries are processed differently: they could be further reprocessed manually or sent to a more advanced LLM. Following previous work on UE in text generation (Malinin and Gales, 2021; Vazhentsev et al., 2022), we compare the methods using the Prediction Rejection Ratio (PRR) metric (Malinin et al., 2017).

Consider a test dataset  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ . Let  $f(\mathbf{x}_i)$  be the output generated by an LLM and  $U(\mathbf{x}_i)$  be the uncertainty score of a prediction. The prediction rejection (PR) curve indicates the dependence of the average quality  $Q(f(\mathbf{x}_i), \mathbf{y}_i)$  of the covered instances from the uncertainty rate  $a$  used for rejection, in ascending order. We use ROUGE-L and BERTScore (Zhang et al., 2020) as text quality metrics  $Q(f(\mathbf{x}_i), \mathbf{y}_i)$ . Finally, PRR computes the ratio of the area  $AUCPR_{unc}$  between the PR curve for the uncertainty estimates and random estimates and the area  $AUCPR_{oracle}$  between the oracle and random estimates:

$$PRR = \frac{AUCPR_{unc}}{AUCPR_{oracle}} \quad (2)$$

Higher PRR values indicate better quality of selective generation.

## 6 Experimental Results

Tables 2 and 3 present the results for Vicuna-v1.5-7b and LLaMA-v2-7b correspondingly.

For both models, the better performance is usually demonstrated by the white-box methods based on information-theoretic concepts (first 8 rows of the table). These methods are in general also easy to implement and computationally lightweight, with the notable exceptions of Semantic Entropy, Monte Carlo Sequence Entropy, and Monte Carlo Normalized Sequence Entropy, which require sampling from the model several times to obtain uncertainty scores.

| UE Method                                | AESLC      |            | XSUM       |            | CoQA       |            | bAbiQA     |           | WMT14 De-En |            | WMT14 Fr-En |            |
|--|------------|------------|------------|------------|------------|------------|------------|-----------|-------------|------------|-------------|------------|
|  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore | ROUGE-L     | BERTScore  | ROUGE-L     | BERTScore  |
| Maximum Sequence Probability             | 0.24±0.01  | 0.19±0.10  | 0.01±0.03  | -0.18±0.15 | 0.35±0.01  | 0.29±0.01  | 0.66±0.02  | 0.82±0.03 | 0.32±0.01   | 0.39±0.01  | 0.25±0.01   | 0.31±0.01  |
| Perplexity                               | 0.19±0.01  | 0.06±0.11  | 0.04±0.03  | -0.13±0.13 | 0.11±0.01  | -0.09±0.02 | 0.65±0.02  | 0.79±0.03 | 0.41±0.01   | 0.47±0.01  | 0.32±0.01   | 0.35±0.01  |
| Mean Token Entropy                       | 0.21±0.01  | 0.08±0.11  | 0.05±0.03  | -0.11±0.13 | 0.10±0.01  | -0.11±0.02 | 0.52±0.02  | 0.68±0.04 | 0.44±0.01   | 0.49±0.01  | 0.35±0.01   | 0.39±0.01  |
| Pointwise Mutual Information             | 0.01±0.01  | -0.01±0.11 | 0.14±0.03  | 0.06±0.13  | -0.24±0.01 | -0.42±0.02 | 0.14±0.03  | 0.55±0.06 | 0.14±0.01   | 0.09±0.01  | 0.11±0.01   | 0.10±0.01  |
| Conditional Pointwise Mutual Information | 0.19±0.01  | 0.06±0.11  | 0.04±0.03  | -0.13±0.14 | 0.11±0.01  | -0.09±0.01 | 0.65±0.02  | 0.79±0.03 | 0.41±0.01   | 0.47±0.01  | 0.32±0.01   | 0.35±0.01  |
| Monte Carlo Sequence Entropy             | 0.22±0.02  | 0.16±0.10  | 0.03±0.03  | -0.14±0.15 | 0.33±0.01  | 0.26±0.01  | 0.65±0.02  | 0.80±0.03 | 0.32±0.01   | 0.39±0.01  | 0.25±0.01   | 0.31±0.01  |
| Monte Carlo Normalized Sequence Entropy  | 0.18±0.01  | 0.06±0.10  | 0.06±0.03  | -0.15±0.13 | 0.09±0.01  | -0.10±0.01 | 0.62±0.02  | 0.68±0.03 | 0.41±0.01   | 0.47±0.01  | 0.31±0.01   | 0.34±0.01  |
| Semantic Entropy                         | 0.22±0.01  | 0.16±0.10  | 0.04±0.03  | -0.11±0.14 | 0.32±0.01  | 0.25±0.01  | 0.65±0.02  | 0.79±0.03 | 0.32±0.01   | 0.39±0.01  | 0.25±0.01   | 0.31±0.01  |
| P(True)                                  | -0.02±0.01 | -0.05±0.11 | 0.12±0.03  | 0.17±0.13  | 0.08±0.01  | 0.09±0.02  | 0.30±0.03  | 0.65±0.05 | -0.00±0.01  | -0.05±0.01 | 0.04±0.01   | -0.02±0.01 |
| Lexical Similarity ROUGE-1               | 0.17±0.01  | 0.15±0.11  | 0.08±0.03  | 0.01±0.13  | 0.17±0.01  | 0.13±0.02  | 0.43±0.03  | 0.58±0.04 | 0.26±0.01   | 0.28±0.01  | 0.14±0.01   | 0.13±0.01  |
| Lexical Similarity ROUGE-L               | 0.17±0.02  | 0.15±0.11  | 0.09±0.03  | 0.00±0.13  | 0.17±0.01  | 0.13±0.01  | 0.43±0.03  | 0.58±0.04 | 0.25±0.01   | 0.28±0.01  | 0.14±0.01   | 0.15±0.01  |
| Lexical Similarity BLEU                  | 0.13±0.01  | 0.08±0.11  | 0.08±0.03  | -0.02±0.13 | 0.14±0.01  | 0.10±0.02  | 0.43±0.03  | 0.56±0.05 | 0.23±0.01   | 0.31±0.01  | 0.13±0.01   | 0.16±0.01  |
| NumSemSets                               | 0.12±0.01  | 0.12±0.11  | 0.04±0.03  | 0.07±0.15  | 0.12±0.01  | 0.08±0.01  | 0.43±0.03  | 0.59±0.05 | 0.03±0.01   | 0.08±0.01  | -0.03±0.01  | -0.00±0.01 |
| EigValLaplacian NLI Score entail.        | 0.16±0.01  | 0.12±0.11  | 0.07±0.03  | 0.02±0.13  | 0.20±0.01  | 0.16±0.01  | 0.32±0.03  | 0.53±0.05 | 0.18±0.01   | 0.24±0.01  | 0.12±0.01   | 0.14±0.01  |
| EigValLaplacian NLI Score contra.        | 0.13±0.01  | 0.13±0.11  | 0.06±0.03  | 0.04±0.13  | 0.18±0.01  | 0.13±0.02  | 0.35±0.03  | 0.45±0.05 | 0.19±0.01   | 0.29±0.01  | 0.09±0.01   | 0.13±0.01  |
| EigValLaplacian Jaccard Score            | 0.13±0.01  | 0.11±0.11  | 0.09±0.03  | -0.00±0.13 | 0.14±0.01  | 0.09±0.01  | 0.43±0.03  | 0.59±0.04 | 0.24±0.01   | 0.31±0.01  | 0.14±0.01   | 0.17±0.01  |
| DegMat NLI Score entail.                 | 0.16±0.02  | 0.15±0.11  | 0.08±0.03  | 0.06±0.13  | 0.14±0.01  | 0.06±0.01  | 0.47±0.03  | 0.55±0.05 | 0.17±0.01   | 0.32±0.01  | 0.18±0.01   | 0.27±0.01  |
| DegMat NLI Score contra.                 | 0.12±0.01  | 0.10±0.11  | 0.13±0.03  | 0.19±0.13  | 0.04±0.01  | -0.07±0.01 | 0.52±0.02  | 0.52±0.04 | 0.18±0.01   | 0.33±0.01  | 0.13±0.01   | 0.25±0.01  |
| DegMat Jaccard Score                     | 0.13±0.02  | 0.11±0.11  | 0.08±0.03  | -0.00±0.13 | 0.15±0.01  | 0.05±0.01  | 0.46±0.03  | 0.58±0.05 | 0.21±0.01   | 0.34±0.01  | 0.16±0.01   | 0.26±0.01  |
| Eccentricity NLI Score entail.           | 0.27±0.01  | 0.18±0.11  | 0.04±0.03  | -0.02±0.13 | 0.35±0.01  | 0.26±0.01  | 0.43±0.02  | 0.63±0.04 | 0.27±0.01   | 0.38±0.01  | 0.18±0.01   | 0.24±0.01  |
| Eccentricity NLI Score contra.           | 0.21±0.01  | 0.16±0.11  | 0.07±0.03  | 0.15±0.14  | 0.19±0.01  | 0.05±0.01  | 0.46±0.03  | 0.58±0.05 | 0.21±0.01   | 0.34±0.01  | 0.16±0.01   | 0.26±0.01  |
| Eccentricity Jaccard Score               | 0.23±0.01  | 0.13±0.10  | 0.07±0.03  | -0.06±0.13 | 0.29±0.01  | 0.19±0.01  | 0.43±0.03  | 0.64±0.05 | 0.35±0.01   | 0.42±0.01  | 0.27±0.01   | 0.32±0.01  |
| Mahalanobis Distance - Decoder           | 0.03±0.01  | -0.01±0.11 | 0.03±0.03  | 0.03±0.15  | 0.03±0.01  | 0.07±0.01  | 0.36±0.03  | 0.57±0.05 | -0.02±0.01  | -0.00±0.01 | -0.02±0.01  | -0.01±0.01 |
| Relative Mahalanobis Distance - Decoder  | 0.02±0.01  | 0.04±0.11  | -0.03±0.03 | -0.07±0.12 | 0.03±0.01  | 0.07±0.02  | -0.25±0.04 | 0.04±0.08 | -0.09±0.01  | -0.08±0.01 | -0.06±0.01  | -0.05±0.01 |
| RDE - Decoder                            | -0.03±0.01 | -0.05±0.11 | 0.07±0.03  | 0.09±0.13  | 0.04±0.01  | 0.09±0.02  | 0.29±0.03  | 0.42±0.06 | -0.01±0.01  | -0.02±0.01 | -0.01±0.01  | -0.02±0.01 |
| HUQ-MD - Decoder                         | 0.19±0.01  | 0.06±0.11  | 0.03±0.03  | -0.10±0.14 | 0.09±0.01  | -0.03±0.02 | 0.62±0.02  | 0.76±0.03 | 0.29±0.01   | 0.36±0.01  | 0.22±0.01   | 0.26±0.01  |
| HUQ-RMD - Decoder                        | 0.19±0.01  | 0.06±0.11  | 0.02±0.03  | -0.13±0.14 | 0.09±0.01  | -0.03±0.01 | 0.31±0.03  | 0.60±0.05 | 0.22±0.01   | 0.26±0.01  | 0.19±0.01   | 0.21±0.01  |

Table 2: PRR↑ for the Vicuna model with ROUGE-L and BERTScore as text quality metrics. Darker color indicates better results.

| UE Method                                | AESLC      |            | XSUM       |            | CoQA       |            | bAbiQA     |            | WMT14 De-En |            | WMT14 Fr-En |            |
|--|------------|------------|------------|------------|------------|------------|------------|------------|-------------|------------|-------------|------------|
|  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore  | ROUGE-L    | BERTScore  | ROUGE-L     | BERTScore  | ROUGE-L     | BERTScore  |
| Maximum Sequence Probability             | 0.22±0.02  | 0.22±0.10  | 0.12±0.03  | 0.16±0.03  | 0.44±0.01  | 0.45±0.01  | 0.43±0.03  | 0.93±0.00  | 0.44±0.01   | 0.64±0.01  | 0.45±0.01   | 0.60±0.02  |
| Perplexity                               | 0.12±0.02  | 0.01±0.10  | 0.13±0.03  | -0.04±0.03 | 0.32±0.01  | 0.18±0.01  | 0.43±0.03  | 0.93±0.00  | 0.43±0.01   | 0.46±0.01  | 0.40±0.01   | 0.41±0.02  |
| Mean Token Entropy                       | 0.13±0.01  | 0.01±0.10  | 0.13±0.04  | -0.06±0.03 | 0.33±0.01  | 0.16±0.01  | 0.43±0.04  | 0.99±0.00  | 0.43±0.01   | 0.42±0.01  | 0.41±0.01   | 0.37±0.02  |
| Pointwise Mutual Information             | -0.07±0.01 | -0.07±0.10 | 0.16±0.04  | 0.05±0.03  | -0.18±0.01 | -0.33±0.02 | -0.35±0.03 | -1.93±0.04 | -0.47±0.01  | -0.91±0.01 | -0.59±0.01  | -0.93±0.04 |
| Conditional Pointwise Mutual Information | 0.12±0.01  | 0.01±0.10  | 0.13±0.04  | -0.04±0.03 | 0.32±0.01  | 0.18±0.01  | 0.43±0.03  | 0.93±0.00  | 0.43±0.01   | 0.46±0.01  | 0.40±0.01   | 0.41±0.02  |
| Monte Carlo Sequence Entropy             | 0.21±0.02  | 0.20±0.09  | 0.13±0.04  | 0.16±0.03  | 0.43±0.01  | 0.44±0.01  | 0.42±0.03  | 0.84±0.01  | 0.41±0.01   | 0.59±0.01  | 0.40±0.01   | 0.52±0.02  |
| Monte Carlo Normalized Sequence Entropy  | 0.14±0.02  | 0.05±0.09  | 0.14±0.03  | -0.01±0.03 | 0.30±0.01  | 0.16±0.01  | 0.37±0.04  | 0.83±0.01  | 0.43±0.01   | 0.47±0.01  | 0.40±0.01   | 0.43±0.02  |
| Semantic Entropy                         | 0.21±0.02  | 0.19±0.09  | 0.13±0.04  | 0.17±0.03  | 0.43±0.01  | 0.44±0.01  | 0.41±0.04  | 0.79±0.01  | 0.40±0.01   | 0.57±0.01  | 0.39±0.01   | 0.51±0.02  |
| P(True)                                  | 0.03±0.01  | 0.09±0.09  | -0.17±0.03 | -0.26±0.04 | -0.08±0.01 | -0.11±0.02 | -0.13±0.03 | 0.98±0.00  | -0.07±0.01  | -0.11±0.01 | -0.02±0.01  | 0.01±0.02  |
| Lexical Similarity ROUGE-1               | 0.18±0.02  | 0.15±0.10  | 0.16±0.03  | 0.13±0.03  | 0.29±0.01  | 0.33±0.01  | 0.15±0.04  | 0.51±0.02  | 0.39±0.01   | 0.52±0.01  | 0.38±0.01   | 0.45±0.02  |
| Lexical Similarity ROUGE-L               | 0.16±0.02  | 0.16±0.10  | 0.16±0.04  | 0.13±0.03  | 0.29±0.01  | 0.33±0.01  | 0.15±0.04  | 0.51±0.02  | 0.38±0.01   | 0.50±0.01  | 0.37±0.01   | 0.47±0.02  |
| Lexical Similarity BLEU                  | 0.13±0.02  | 0.09±0.10  | 0.15±0.04  | 0.08±0.03  | 0.26±0.01  | 0.25±0.01  | 0.25±0.03  | 0.63±0.01  | 0.39±0.01   | 0.50±0.01  | 0.37±0.01   | 0.47±0.02  |
| NumSemSets                               | 0.08±0.01  | 0.08±0.10  | 0.03±0.03  | 0.10±0.03  | 0.21±0.01  | 0.20±0.02  | 0.19±0.04  | 0.51±0.02  | 0.05±0.01   | 0.06±0.01  | -0.02±0.01  | 0.01±0.03  |
| EigValLaplacian NLI Score entail.        | 0.19±0.01  | 0.17±0.09  | 0.10±0.03  | 0.22±0.03  | 0.27±0.01  | 0.28±0.01  | 0.04±0.03  | 0.71±0.01  | 0.32±0.01   | 0.44±0.01  | 0.29±0.01   | 0.37±0.02  |
| EigValLaplacian NLI Score contra.        | 0.15±0.02  | 0.13±0.10  | 0.08±0.03  | 0.20±0.03  | 0.26±0.01  | 0.28±0.01  | 0.08±0.04  | 0.67±0.01  | 0.32±0.01   | 0.44±0.01  | 0.28±0.01   | 0.38±0.02  |
| EigValLaplacian Jaccard Score            | 0.15±0.02  | 0.12±0.10  | 0.16±0.04  | 0.13±0.03  | 0.26±0.01  | 0.22±0.01  | 0.21±0.03  | 0.67±0.02  | 0.40±0.01   | 0.54±0.01  | 0.39±0.01   | 0.51±0.02  |
| DegMat NLI Score entail.                 | 0.16±0.01  | 0.16±0.09  | 0.11±0.04  | 0.23±0.03  | 0.12±0.01  | 0.00±0.01  | 0.06±0.03  | -0.13±0.03 | 0.34±0.01   | 0.50±0.01  | 0.33±0.01   | 0.46±0.02  |
| DegMat NLI Score contra.                 | 0.07±0.01  | 0.06±0.10  | 0.09±0.03  | 0.23±0.03  | -0.03±0.01 | -0.15±0.01 | 0.12±0.04  | -0.17±0.03 | 0.33±0.01   | 0.53±0.01  | 0.34±0.01   | 0.50±0.02  |
| DegMat Jaccard Score                     | 0.15±0.01  | 0.11±0.10  | 0.16±0.03  | 0.12±0.03  | 0.27±0.01  | 0.24±0.01  | 0.25±0.04  | 0.63±0.02  | 0.42±0.01   | 0.55±0.01  | 0.39±0.01   | 0.50±0.02  |
| Eccentricity NLI Score entail.           | 0.21±0.01  | 0.18±0.10  | 0.09±0.03  | 0.22±0.03  | 0.43±0.01  | 0.42±0.01  | 0.11±0.04  | 0.74±0.01  | 0.30±0.01   | 0.41±0.01  | 0.23±0.01   | 0.29±0.02  |
| Eccentricity NLI Score contra.           | 0.15±0.01  | 0.11±0.10  | 0.06±0.03  | 0.13±0.03  | 0.36±0.01  | 0.32±0.01  | 0.38±0.04  | 0.32±0.03  | 0.22±0.01   | 0.33±0.01  | 0.19±0.01   | 0.28±0.02  |
| Eccentricity Jaccard Score               | 0.18±0.02  | 0.15±0.09  | 0.15±0.03  | 0.06±0.03  | 0.42±0.01  | 0.42±0.01  | 0.19±0.04  | 0.66±0.01  | 0.43±0.01   | 0.50±0.01  | 0.41±0.01   | 0.46±0.02  |
| Mahalanobis Distance - Decoder           | 0.00±0.01  | -0.01±0.10 | 0.00±0.03  | 0.17±0.03  | -0.02±0.01 | 0.06±0.01  | 0.31±0.04  | -0.30±0.03 | -0.07±0.01  | -0.10±0.01 | -0.14±0.01  | -0.21±0.03 |
| Relative Mahalanobis Distance - Decoder  | 0.03±0.01  | 0.05±0.09  | -0.10±0.03 | -0.24±0.03 | -0.04±0.01 | 0.05±0.01  | -0.25±0.03 | 0.24±0.02  | 0.01±0.01   | 0.10±0.01  | 0.17±0.01   | 0.30±0.02  |
| RDE - Decoder                            | -0.05±0.01 | -0.06±0.10 | 0.04±0.03  | 0.23±0.03  | -0.01±0.01 | 0.08±0.01  | 0.30±0.04  | -0.29±0.03 | -0.06±0.01  | -0.08±0.01 | -0.08±0.01  | -0.15±0.03 |
| HUQ-MD - Decoder                         | 0.07±0.01  | -0.01±0.10 | 0.13±0.03  | -0.04±0.03 | 0.30±0.01  | 0.17±0.01  | 0.43±0.04  | 0.93±0.00  | 0.21±0.01   | 0.19±0.01  | 0.13±0.01   | 0.06±0.03  |
| HUQ-RMD - Decoder                        | 0.11±0.02  | 0.04±0.10  | 0.13±0.03  | -0.04±0.03 | 0.30±0.01  | 0.17±0.01  | 0.43±0.03  | 0.93±0.00  | 0.25±0.01   | 0.30±0.01  | 0.35±0.01   | 0.42±0.02  |

Table 3: PRR↑ for the LLaMA-2 model with ROUGE-L and BERTScore as text quality metrics. Darker color indicates better results.

When working with LLMs as web services, usually there is no access to full posterior distributions over tokens, therefore, only black-box methods could be used. Among this group of approaches, the best average performance is achieved by Eccentricity for Vicuna. For LLaMA, there is no clear advantage for any of the methods considered.

Overall, we see that absolute values for all evaluated methods, models, and datasets are far away from perfect. Low performance of current methods is especially evident on more complicated tasks such as XSum and WMT14. Our experimental results demonstrate that the task of selective generation is not close to be solved. This once again underlines the importance of further research and development of efficient uncertainty estimation techniques for generative language models.

## 7 Conclusion

As the community strives to advance the potential of LLMs, it is critical to be mindful about dangers of their uncontrolled usage. In this work, we propose a tool for making the application of LLMs safer. Enriching model predictions with uncertainty scores helps users and developers to be informed about these risks, encouraging healthy skepticism towards certain outputs generated by these models.

We plan to further expand our framework with implementations of new UE methods that emerge in the future. We hope that our work will foster the development of techniques to detect and mitigate LLM hallucinations, which we believe is a key to unlocking the safe, responsible, and effective use of LLMs in real-world applications.

## Limitations

We have tried to be as comprehensive as possible with our collection of UE methods. However, we omit several techniques that have not demonstrated strong performance in previous work, do not have a strong theoretical motivation, or are similar to other implemented techniques.

We note that comprehensive evaluation of UE methods is an open research question. LM-Polygraph makes the first steps to systematize, and provide interfaces and tools for testing UE techniques in a unified manner. However, we believe that the number of tasks and datasets should be extended in the future.

When running the demo, we cannot provide an access to the biggest and the most powerful public LLMs, because running them is prohibitively expensive. Nevertheless, a user can access models such as ChatGPT by providing an API access key.

LM-Polygraph supports common application program interfaces used by modern LLMs. However, it is possible that certain modifications will be required to support future releases of LLMs.

At the moment of writing, LM-polygraph provides valid uncertainty estimates only for model outputs in English language. This is due to the fact that most generation quality metrics implemented are based off English-specific implementations and non-multilingual models. We plan to alleviate this limitation by allowing the user to easily employ custom quality metrics and scoring models.

## Ethics Statement

We conducted all experiments on publicly-available datasets that have been leveraged in various previous work on uncertainty estimation of LLMs.

While training data for most LLMs, such as BLOOMz, was selected to contain little or no abusive text content, such models can still potentially output harmful textual content. Techniques investigated in our work estimate certainty of an LM output to “censor” its output, and model debiasing is an orthogonal direction to our line of work. These additional methods can and perhaps should be combined in real production LLM deployments. We hope that our framework contributes to safer and more reliable usage of language models.

## Acknowledgements

We thank the anonymous reviewers for their insightful feedback towards improving this paper.

## References

- Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. 2014. [Findings of the 2014 workshop on statistical machine translation](#). In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. 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.
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. [Free dolly: Introducing the world’s first truly open instruction-tuned llm](#).
- Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. [Evaluating prerequisite qualities for learning end-to-end dialog systems](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. 2022. [On the origin of hallucinations in conversational models: Is it the datasets or the models?](#) In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5271–5285, Seattle, United States. Association for Computational Linguistics.
- Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. [Unsupervised quality estimation for neural machine translation](#). *Transactions of the Association for Computational Linguistics*, 8:539–555.
- Yarin Gal. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.
- Jianfeng He, Xuchao Zhang, Shuo Lei, Zhiqian Chen, Fanglan Chen, Abdulaziz Alhamadani, Bei Xiao, and Chang-Tien Lu. 2020. [Towards more accurate uncertainty estimation in text classification](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8362–8372. Association for Computational Linguistics.



- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [Deberta: decoding-enhanced bert with disentangled attention](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. [Language models \(mostly\) know what they know](#). *CoRR*, abs/2207.05221.
- Nikita Kotelevskii, Aleksandr Artemenkov, Kirill Fedyanin, Fedor Noskov, Alexander Fishkov, Artem Shelmanov, Artem Vazhentsev, Aleksandr Petiushko, and Maxim Panov. 2022. [Nonparametric uncertainty quantification for single deterministic neural network](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 36308–36323. Curran Associates, Inc.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. [Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Gleb Kuzmin, Artem Vazhentsev, Artem Shelmanov, Xudong Han, Simon Suster, Maxim Panov, Alexander Panchenko, and Timothy Baldwin. 2023. [Uncertainty estimation for debiased models: Does fairness hurt reliability?](#) In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 744–770, Nusa Dua, Bali. Association for Computational Linguistics.
- Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. [A simple unified framework for detecting out-of-distribution samples and adversarial attacks](#). In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, volume 31, pages 7167–7177.
- Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. [Generating with confidence: Uncertainty quantification for black-box large language models](#). *CoRR*, abs/2305.19187.
- Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. [ParaDetox: Detoxification with parallel data](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6804–6818, Dublin, Ireland. Association for Computational Linguistics.
- Andrey Malinin and Mark J. F. Gales. 2021. [Uncertainty estimation in autoregressive structured prediction](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Andrey Malinin, Anton Ragni, Kate Knill, and Mark Gales. 2017. [Incorporating uncertainty into deep learning for spoken language assessment](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–50, Vancouver, Canada. Association for Computational Linguistics.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.
- 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.
- 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, Danqi Chen, and Christopher D. Manning. 2019. [CoQA: A conversational question answering challenge](#). *Transactions of the Association for Computational Linguistics*, 7:249–266.
- Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. 2023. [Out-of-distribution detection and selective generation for conditional language models](#). In *The Eleventh International Conference on Learning Representations*.
- Peter J Rousseeuw. 1984. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie 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 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, and et al. 2022. [BLOOM: A 176b-parameter open-access multilingual language model](#). *CoRR*, abs/2211.05100.
- Artem Shelmanov, Evgenii Tsymbalov, Dmitri Puzyrev, Kirill Fedyanin, Alexander Panchenko, and Maxim Panov. 2021. [How certain is your Transformer?](#) In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1833–1840, Online. Association for Computational Linguistics.
- Junya Takayama and Yuki Arase. 2019. [Relevant and informative response generation using pointwise mutual information](#). In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 133–138, Florence, Italy. Association for Computational Linguistics.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [Stanford alpaca: An instruction-following llama model](#).
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *CoRR*, abs/2307.09288.
- Liam van der Poel, Ryan Cotterell, and Clara Meister. 2022. [Mutual information alleviates hallucinations in abstractive summarization](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5956–5965, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Artem Vazhentsev, Gleb Kuzmin, Artem Shelmanov, Akim Tsvigun, Evgenii Tsymbalov, Kirill Fedyanin, Maxim Panov, Alexander Panchenko, Gleb Gusev, Mikhail Burtsev, Manvel Avetisian, and Leonid Zhukov. 2022. [Uncertainty estimation of transformer predictions for misclassification detection](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8237–8252, Dublin, Ireland. Association for Computational Linguistics.
- Artem Vazhentsev, Gleb Kuzmin, Akim Tsvigun, Alexander Panchenko, Maxim Panov, Mikhail Burtsev, and Artem Shelmanov. 2023a. [Hybrid uncertainty quantification for selective text classification in ambiguous tasks](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11659–11681, Toronto, Canada. Association for Computational Linguistics.
- Artem Vazhentsev, Akim Tsvigun, Roman Vashurin, Sergey Petrakov, Daniil Vasilev, Maxim Panov, Alexander Panchenko, and Artem Shelmanov. 2023b. [Efficient out-of-domain detection for sequence to sequence models](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1430–1454, Toronto, Canada. Association for Computational Linguistics.
- Yuxia Wang, Daniel Beck, Timothy Baldwin, and Karin Verspoor. 2022. Uncertainty estimation and reduction of pre-trained models for text regression. *Transactions of the Association for Computational Linguistics*, 10:680–696.
- Yijun Xiao and William Yang Wang. 2021. [On hallucination and predictive uncertainty in conditional language generation](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2734–2744, Online. Association for Computational Linguistics.
- Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. [The art of abstention: Selective prediction and error regularization 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 (Volume 1: Long Papers)*, pages 1040–1051, Online. Association for Computational Linguistics.
- Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna J. Martindale, and Marine Carpuat. 2023. [Understanding and detecting hallucinations in neural machine translation via model introspection](#). *Transactions of the Association for Computational Linguistics*, 11:546–564.

- Zheng Xin Yong, Hailey Schoelkopf, Niklas Muenighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M. Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, Genta Indra Winata, Stella Biderman, Edward Raff, Dragomir Radev, and Vassilina Nikoulina. 2023. [BLOOM+1: adding language support to BLOOM for zero-shot prompting](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2023, Toronto, Canada, July 9-14, 2023, pages 11682–11703. Association for Computational Linguistics.
- KiYoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. 2022. [Detection of adversarial examples in text classification: Benchmark and baseline via robust density estimation](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3656–3672, Dublin, Ireland. Association for Computational Linguistics.
- Rui Zhang and Joel R. Tetreault. 2019. [This email could save your life: Introducing the task of email subject line generation](#). 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 446–456. Association for Computational Linguistics.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with BERT](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Xuchao Zhang, Fanglan Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2019. [Mitigating uncertainty in document classification](#). 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 3126–3136, Minneapolis, Minnesota. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). *CoRR*, abs/2306.05685.

## A Methods Description

Here, we summarize UE methods implemented in LM-Polygraph; see also Table 1.

### A.1 White-box Methods

#### A.1.1 Information-based methods

*Maximum sequence probability* score simply leverages the probability of the most likely sequence generation:  $\text{MSP}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = 1 - P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ .

*Length-normalized log probability* computes the average negative log probability of generated tokens. If the score is exponentiated it corresponds to *perplexity*. The resulting quantity is computed as

$$P(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}) = \exp\left\{-\frac{1}{L} \log P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})\right\},$$

while it is convenient also to denote length-normalized sequence probability by  $\bar{P}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \exp\left\{\frac{1}{L} \log P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})\right\}$ .

We also provide the *mean token entropy*, where we simply average entropy of each individual token in the generated sequence:

$$\mathcal{H}_T(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{L} \sum_{l=1}^L \mathcal{H}(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}),$$

where  $\mathcal{H}(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta})$  is an entropy of the token distribution  $P(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta})$ .

The other possibility to compute entropy-based uncertainty measure is to compute it on the level of whole sequences via  $\mathbb{E}[-\log P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})]$ , where expectation is taken over the sequences  $\mathbf{y}$  randomly generated from the distribution  $P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ . In practice, one needs to use Monte-Carlo integration, i.e. generate several sequences  $\mathbf{y}^{(k)}$ ,  $k = 1, \dots, K$  via randoms sampling and compute the resulting *Monte Carlo Sequence Entropy*:

$$\mathcal{H}_S(\mathbf{x}; \boldsymbol{\theta}) = -\frac{1}{K} \sum_{k=1}^K \log P(\mathbf{y}^{(k)} \mid \mathbf{x}, \boldsymbol{\theta}). \quad (3)$$

The same procedure can be done by substituting  $P(\mathbf{y}^{(k)} \mid \mathbf{x}, \boldsymbol{\theta})$  with its length-normalized version  $\bar{P}(\mathbf{y}^{(k)} \mid \mathbf{x}, \boldsymbol{\theta})$  leading to a more reliable uncertainty measure in some applications.

Another entropy-based uncertainty measure is *Semantic Entropy* proposed by Kuhn et al. (2023). The method aims to deal with the generated sequences that have similar meaning while having different probabilities according to the model, which can significantly affect the resulting entropy

value (3). The idea is to cluster generated sequences  $\mathbf{y}^{(k)}$ ,  $k = 1, \dots, K$  into several semantically homogeneous clusters  $\mathcal{C}_m$ ,  $m = 1, \dots, M$  with  $M \leq K$  with bi-directional entailment algorithm and average the sequence probabilities within the clusters. The resulting estimate of entropy is given by the following formula:

$$\text{SE}(\mathbf{x}; \boldsymbol{\theta}) = -\sum_{m=1}^M \hat{P}_m(\mathbf{x}; \boldsymbol{\theta}) \log \hat{P}_m(\mathbf{x}; \boldsymbol{\theta}),$$

where  $\hat{P}_m(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{|\mathcal{C}_m|} \sum_{\mathbf{y} \in \mathcal{C}_m} P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ .

Finally, one can consider negative mean *Pointwise Mutual Information* (PMI; Takayama and Arase (2019)) which is given by

$$\text{PMI}(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{L} \sum_{l=1}^L \log \frac{P(y_l \mid \mathbf{y}_{<l}, \boldsymbol{\theta})}{P(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta})}.$$

This method was extended in (van der Poel et al., 2022) by considering only those marginal probabilities for which the entropy of the conditional distribution is above certain threshold:  $\mathcal{H}(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}) \geq \tau$ . It leads to the negative mean *Conditional Pointwise Mutual Information* (CPMI) measure that is given by:

$$\begin{aligned} \text{CPMI}(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}) = & -\frac{1}{L} \sum_{l=1}^L \log P(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}) \\ & + \frac{\lambda}{L} \sum_{l: \mathcal{H}(y_l \mid \mathbf{y}_{<l}, \mathbf{x}, \boldsymbol{\theta}) \geq \tau} \log P(y_l \mid \mathbf{y}_{<l}, \boldsymbol{\theta}), \end{aligned}$$

where  $\lambda > 0$  is another tunable parameter.

#### A.1.2 Ensemble-based methods

For the ensembling on a sequence level, we consider two uncertainty measures: total uncertainty measured via average sequence probability  $\bar{P}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{M} \sum_{i=1}^M \bar{P}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_i)$ :

$$\text{MSP}_S(\mathbf{y}, \mathbf{x}) = 1 - \bar{P}(\mathbf{y} \mid \mathbf{x}) \quad (4)$$

and

$$\mathcal{M}_S(\mathbf{y}, \mathbf{x}) = \frac{1}{M} \sum_{i=1}^M \log \frac{P(\mathbf{y} \mid \mathbf{x})}{P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_i)}, \quad (5)$$

which is known as *reverse mutual information* (RMI).

Next we discuss token level uncertainty measures and start with a total uncertainty estimate via entropy:

$$\mathcal{H}_T(\mathbf{y}, \mathbf{x}) = \sum_{l=1}^L \mathcal{H}(y_l \mid \mathbf{y}_{<l}, \mathbf{x}), \quad (6)$$

where  $\mathcal{H}(y_l | \mathbf{y}_{<l}, \mathbf{x})$  is an entropy of the token distribution  $P(y_l | \mathbf{y}_{<l}, \mathbf{x}) = \frac{1}{M} \sum_{i=1}^M P(y_l | \mathbf{y}_{<l}, \mathbf{x}; \theta_i)$ .

Additionally, for the ensemble one can compute the variety of other token level uncertainty measures including average entropy of ensemble members (also known as *Data Uncertainty*):

$$\mathcal{D}(y_l | \mathbf{y}_{<l}, \mathbf{x}) = \frac{1}{M} \sum_{i=1}^M \mathcal{H}(y_l | \mathbf{y}_{<l}, \mathbf{x}, \theta_i),$$

*Mutual Information (MI)*:

$$\mathcal{I}(y_l | \mathbf{y}_{<l}, \mathbf{x}) = \mathcal{H}(y_l | \mathbf{y}_{<l}, \mathbf{x}) - \mathcal{D}(y_l | \mathbf{y}_{<l}, \mathbf{x})$$

and *Expected Pairwise KL Divergence (EPKL)*:

$$\mathcal{K}(y_l | \mathbf{y}_{<l}, \mathbf{x}) = \binom{M}{2}^{-1} \cdot \sum_{i \neq j} \mathcal{K}\mathcal{L}(P(y_l | \mathbf{y}_{<l}, \mathbf{x}, \theta_i) \| P(y_l | \mathbf{y}_{<l}, \mathbf{x}, \theta_j)),$$

where  $\mathcal{K}\mathcal{L}(P \| Q)$  refers to a KL-divergence between distributions  $P$  and  $Q$ .

Finally, *Reverse Mutual Information (RMI)* also can be computed on the token level via a simple equation

$$\mathcal{M}(y_l | \mathbf{y}_{<l}, \mathbf{x}) = \mathcal{K}(y_l | \mathbf{y}_{<l}, \mathbf{x}) - \mathcal{I}(y_l | \mathbf{y}_{<l}, \mathbf{x}).$$

The resulting token-level uncertainties computed via Data Uncertainty, MI, EPKL and RMI can be plugged-in in equation (6) on the place of entropy leading to corresponding sequence level uncertainty estimates.

### A.1.3 Density-based Methods

Let  $h(\mathbf{x})$  be a hidden representation of an instance  $\mathbf{x}$ . The *Mahalanobis Distance* (MD; Lee et al. (2018)) method fits a Gaussian centered at the training data centroid  $\mu$  with an empirical covariance matrix  $\Sigma$ . The uncertainty score is the Mahalanobis distance between  $h(\mathbf{x})$  and  $\mu$ :

$$\text{MD}(\mathbf{x}) = (h(\mathbf{x}) - \mu)^T \Sigma^{-1} (h(\mathbf{x}) - \mu).$$

We suggest using the last hidden state of the encoder averaged over non-padding tokens or the last hidden state of the decoder averaged over all generated tokens as  $h(\mathbf{x})$ .

The Robust Density Estimation (RDE; Yoo et al. (2022)) method improves over MD by reducing the dimensionality of  $h(\mathbf{x})$  via PCA decomposition. Additionally, computing of the covariance

matrix  $\Sigma$  for each individual class is done by using the Minimum Covariance Determinant estimation (Rousseeuw, 1984). The uncertainty score is computed as the Mahalanobis distance between but in the space of reduced dimensionality.

Ren et al. (2023) showed that it might be useful to adjust the Mahalanobis distance score by subtracting from it the other Mahalanobis distance  $\text{MD}_0(\mathbf{x})$  computed for some large general purpose dataset covering many domains like C4 (Raffel et al., 2020). The resulting resulting *Relative Mahalanobis Distance* score is

$$\text{RMD}(\mathbf{x}) = \text{MD}(\mathbf{x}) - \text{MD}_0(\mathbf{x}).$$

## A.2 Black-box Methods

In this work, we follow Lin et al. (2023) and consider two approaches to compute the similarity for the generated responses. The first one is *Jaccard similarity*:

$$s(\mathbf{y}, \mathbf{y}') = \frac{|\mathbf{y} \cap \mathbf{y}'|}{|\mathbf{y} \cup \mathbf{y}'|},$$

where the sequences  $\mathbf{y}$  and  $\mathbf{y}'$  are considered just as sets of words.

The other similarity measure considered is Natural Language Index (NLI) which employs a classification model to identify whether two responses are similar. We follow Kuhn et al. (2023) and use the DeBERTa-large model (He et al., 2021) that, for each pair of input sequences, provides two probabilities:  $\hat{p}_{\text{entail}}(\mathbf{y}, \mathbf{y}')$  that measures the degree of entailment between the sequences and  $\hat{p}_{\text{contra}}(\mathbf{y}, \mathbf{y}')$  that measures the contradiction between them. Then one can use  $s_{\text{entail}}(\mathbf{y}, \mathbf{y}') = \hat{p}_{\text{entail}}(\mathbf{y}, \mathbf{y}')$  or  $s_{\text{contra}}(\mathbf{y}, \mathbf{y}') = 1 - \hat{p}_{\text{contra}}(\mathbf{y}, \mathbf{y}')$  as a measure of similarity between sequences  $\mathbf{y}$  and  $\mathbf{y}'$ .

*Number of Semantic Sets* illustrates whether answers are semantically equivalent. We adopt an iterative approach by sequentially examining responses from the first to the last while making pairwise comparisons between them (each pair has indexes  $j_1$  and  $j_2$ ,  $j_2 > j_1$ ). The number of semantic sets initially equals the total number of generated answers  $K$ . If the condition  $\hat{p}_{\text{entail}}(\mathbf{y}_{j_1}, \mathbf{y}_{j_2}) > \hat{p}_{\text{contra}}(\mathbf{y}_{j_1}, \mathbf{y}_{j_2})$  and  $\hat{p}_{\text{entail}}(\mathbf{y}_{j_2}, \mathbf{y}_{j_1}) > \hat{p}_{\text{contra}}(\mathbf{y}_{j_2}, \mathbf{y}_{j_1})$  is fulfilled we put this two sentences into one cluster. The computation is done for all the pairs of answers, and then the resulting number of distinct sets  $U_{\text{NumSemSets}}$

is reported. It is worth noting that a higher number of semantic sets corresponds to an increased level of uncertainty, as it suggests a higher number of diverse semantic interpretations for the answer.

Nonetheless, it is essential to acknowledge a limitation of this measure: it can only take integer values. Additionally, it cannot be assumed that the semantic equivalence derived from the NLI model is always transitive. Consequently, the authors of (Lin et al., 2023) suggest the consideration of a continuous counterpart of this metric. They propose the *Sum of Eigenvalues of the Graph Laplacian* as a potential alternative approach.

Let’s consider a similarity matrix  $S_{j_1 j_2} = (s(\mathbf{y}_{j_1}, \mathbf{y}_{j_2}) + s(\mathbf{y}_{j_2}, \mathbf{y}_{j_1}))/2$ . Averaging is done to obtain better consistency. Normalized Graph Laplacian of the obtained similarity Matrix  $S$  has the following formula  $L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$ , where  $D$  is a diagonal matrix and  $D_{ii} = \sum_{j=1}^K S_{ij}$ . Consequently, the following formula is derived:  $U_{EigV} = \sum_{k=1}^K \max(0, 1 - \lambda_k)$ . This value is a continuous analogue of  $U_{NumSemSets}$ . In extreme case if adjacency matrix  $S$  is binary these two measures will coincide.

Of course, from a theoretical and practical point of view,  $U_{EigV}$  is a much more flexible approach compared to  $U_{NumSemSets}$ . Still, they have a common disadvantage: they can not provide uncertainty for each answer. However, authors of (Lin et al., 2023) demonstrate that we can take it from *Degree Matrix*  $D$  computed above. The idea is that the total uncertainty of the answers might be measured as a corrected trace of the diagonal matrix  $D$  because elements on the diagonal of matrix  $D$  are sums of similarities between the given answer and other answers. Thus, it is an average pairwise distance between all answers, and a larger value will indicate larger uncertainty because of the larger distance between answers. The resulting uncertainty measure becomes  $U_{Deg} = 1 - trace(D)/K^2$ .

A drawback of previously considered methods is the limited knowledge of the actual embedding space for the different answers since we only have measures of their similarities. Nevertheless, we can overcome this limitation by taking advantage of the inferential capabilities of the graph Laplacian, which makes it easier to obtain the coordinates of the answers. Let us introduce  $\mathbf{u}_1, \dots, \mathbf{u}_k \in R^K$  as the eigenvectors of  $L$  that correspond to  $k$  smallest eigenvalues. We can efficiently construct an informative embedding  $\mathbf{v}_j = [\mathbf{u}_{1,j}, \dots, \mathbf{u}_{k,j}]$  for

an answer  $\mathbf{y}_j$ . Authors of (Lin et al., 2023) demonstrate that this approach allows the usage of the average distance from the center as an uncertainty metric and to consider the distance of each response from the center as a measure of (negative) confidence. In mathematical terms, the estimates for *Eccentricity* can be defined as follows:  $U_{Ecc} = \|\tilde{\mathbf{v}}_1^T, \dots, \tilde{\mathbf{v}}_K^T\|_2$ , where  $\tilde{\mathbf{v}}_j = \mathbf{v}_j - \frac{1}{K} \sum_{\ell=1}^K \mathbf{v}_\ell$ .

Last but not least, *Lexical Similarity* is a measure proposed by (Fomicheva et al., 2020) that computes how similar two words or phrases are in terms of their meaning. Since the original article is dedicated to machine translation, this measure calculates the average similarity score between all pairs of translation hypotheses in a set, using a similarity measure based on the overlap of their lexical items. Different metrics can be used, such as ROUGE-1, ROUGE-2, ROUGE-L, and BLEU. For our task, this measure iterates over all responses and calculates the average score with other answers.

## B Generation Hyperparameters

| Dataset     | Task | Max Input Length | Generation Length | Temperature | Top-p | Do Sample | Beams | Repetition Penalty |
|-------------|------|------------------|-------------------|-------------|-------|-----------|-------|--------------------|
| AESLC       | ATS  | 2048             | 31                | 1.0         | 1.0   | False     | 1     | 1                  |
| XSUM        |      |                  | 56                |             |       |           |       |                    |
| CoQA        | 20   |                  |                   |             |       |           |       |                    |
| bAbiQA      | 3    |                  |                   |             |       |           |       |                    |
| WMT14 De-En | 107  |                  |                   |             |       |           |       |                    |
| WMT14 Fr-En | 107  |                  |                   |             |       |           |       |                    |

Table 4: Text generation hyperparameters for both LLMs Vicuna-v1.5-7b and Llama-2-7b used in the experiments.

Table 4 presents the hyperparameters used for experiments with LLMs Vicuna-v1.5-7b and LLaMA-2-7b-hf on various datasets and tasks. Maximum length of generated sequence was set for each dataset as the 99th percentile of target sequence length on the respective train set.

## C Text Generation Quality Metrics

| AESLC   |           | XSUM    |           | CoQA    |           | bAbiQA  |           | WMT14 De-En |           | WMT14 Fr-En |           |
|---------|-----------|---------|-----------|---------|-----------|---------|-----------|-------------|-----------|-------------|-----------|
| Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L     | BERTScore | Rouge-L     | BERTScore |
| 0.24    | 0.83      | 0.18    | 0.86      | 0.29    | 0.85      | 0.68    | 1.0       | 0.59        | 0.95      | 0.64        | 0.95      |

Table 5: Rouge-L $\uparrow$  and BERTScore $\uparrow$  for Vicuna v1.5 model for various tasks.

| AESLC   |           | XSUM    |           | CoQA    |           | bAbiQA  |           | WMT14 De-En |           | WMT14 Fr-En |           |
|---------|-----------|---------|-----------|---------|-----------|---------|-----------|-------------|-----------|-------------|-----------|
| Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L | BERTScore | Rouge-L     | BERTScore | Rouge-L     | BERTScore |
| 0.23    | 0.84      | 0.19    | 0.86      | 0.51    | 0.91      | 0.36    | 0.98      | 0.54        | 0.93      | 0.56        | 0.92      |

Table 6: Rouge-L $\uparrow$  and BERTScore $\uparrow$  for the Llama v2 model for various tasks.

## D Dataset Statistics

Table 7 illustrates the statistics of the datasets that were used in the experiments. Experiments were conducted using all examples from the test sets of these datasets, while training density-based methods were performed on a random subset of 1000 elements from the train set.

```
HYDRA_CONFIG=/path/to/cloned/repo/examples/configs/polygraph_eval_coqa.yaml polygraph_eval model=lmsys/vicuna-7b-v1.5
```

Figure 3: Script that reproduces benchmark results for CoQA dataset with Vicuna-v1.5-7b model.

To evaluate the performance of considered uncertainty estimation methods, we provide code to retrieve benchmark results. Figure 3 shows an example of starting an experiment with the Vicuna-v1.5-7b model on the Questions Answering task (CoQA dataset).

Figure 4 shows an example of a config file used for experiment related to CoQA dataset with Vicuna-v1.5-7b model. It contains information about import and parameters. For other datasets and models the config structure is the same.

## E Normalization of Uncertainty Estimates in Demo App

To make uncertainty estimation more intuitive for the end user, directly interacting with the LLM, we perform normalization of various uncertainty estimates. After normalization the output  $UE(\mathbf{x})$  of any uncertainty estimation approach becomes a confidence score  $C(\mathbf{x}) \in [0, 1] \subset \mathbb{R}$ .

We experimented with several ways of achieving this normalization, including quantile-based approach and simple linear normalization on maximum value obtained from validation dataset. Eventually we

| Dataset    | Num. instances                | Av. document len. | Av. target len. | Language          |
|------------|-------------------------------|-------------------|-----------------|-------------------|
| <b>NMT</b> |                               |                   |                 |                   |
| WMT'14     | 4.51M / 3000 / <b>3003</b>    | 19.8 / 18.3       | 23.0 / 21.3     | German-to-English |
| WMT'14     | 40.8M / 3000 / <b>3003</b>    | 33.5 / 32.1       | 29.2 / 27.0     | French-to-English |
| <b>ATS</b> |                               |                   |                 |                   |
| XSum       | 204045 / 11332 / <b>11334</b> | 454.6             | 26.1            | English           |
| AESLC      | 14436 / 1960 / <b>1906</b>    | 165.5             | 6.7             | English           |
| <b>QA</b>  |                               |                   |                 |                   |
| CoQA       | 7199 / 500 / -                | 271.4             | 2.7             | English           |
| bAbiQA     | 2000 / - / <b>200</b>         | 31.1              | 1.0             | English           |

Table 7: Quantitative information regarding the datasets from experiments. It includes the count of instances available for the training, validation, and **test** sets, as well as the mean lengths of both texts and targets (answers / translations / summaries) measured in terms of tokens. In addition, the languages of the source and target texts are also specified.

```

hydra:
  run:
    dir: ${cache_path}/${task}/${model}/${dataset}/${now:%Y-%m-%d}/${now:%H-%M-%S}

cache_path: ./workdir/output
save_path: '${hydra:run.dir}'

device: cpu

task: qa

dataset: coqa
text_column: questions
label_column: answers
prompt: "Answer a question given a story. Output only the answer.\nStory:\n{story}\n\nQuestion:\n{question}\n\nAnswer:\n"
train_split: train
eval_split: validation
max_new_tokens: 20
load_from_disk: false

train_dataset: null
train_test_split: false
test_split_size: 1

background_train_dataset: allenai/c4
background_train_dataset_text_column: text
background_train_dataset_label_column: url
background_train_dataset_data_files: en/c4-train.00000-of-01024.json.gz
background_load_from_disk: false

subsample_background_train_dataset: 1000
subsample_train_dataset: 1000
subsample_eval_dataset: -1

model: lmsys/vicuna-7b-v1.5
use_auth_token:

use_density_based_ue: true
use_seq_ue: true
use_tok_ue: false

ignore_exceptions: false

batch_size: 1
deberta_batch_size: 10

seed:
  - 1

```

Figure 4: Config Example for Question Answering on CoQA dataset.

performed normalization as a calibration procedure, where normalized confidence score represents expected value of generation quality metric of choice (i.e. RougeL) for a given uncertainty estimate. This expectation is estimated by computing sample averages of quality metric over bins of uncertainty estimates, calculated for some validation dataset. For RougeL metric, the confidence estimate  $C(x_{input})$  thus becomes:



$$C(\mathbf{x}_{input}) = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x}_i, \mathbf{y}_i \in \mathcal{B}} rougeL(\hat{\mathbf{y}}_i, \mathbf{y}_i),$$

where  $\hat{\mathbf{y}}_i$  is model output for input  $\mathbf{x}_i$ , and

$$\mathcal{B} = \{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{calib} \mid UE(\mathbf{x}) \in [UE_{min}, UE_{max}]\}$$

is the bin to which uncertainty estimate of the input belongs. The bounds of this bin are selected from the predetermined set of bin boundaries to be the neighboring pair for which condition

$$UE_{min} \leq UE(\mathbf{x}_{input}) < UE_{max}$$

is satisfied.

This dataset  $\mathcal{D}_{calib}$  is constructed to be representative of different modes of operation of a given model. For this purpose it is constructed as a mixture of several different datasets for different tasks, with different values of relevant statistics, such as input sequence length, typical generated output length etc.

It is obvious that quality of this normalized confidence score depends heavily on the size and diversity of the calibration dataset. In general we consider the problem of translating opaque uncertainty estimates into intuitive absolute confidence scores, that correctly represent likelihood of the generated output being correct and relevant, as an important and complicated task. We leave solving this problem in a more efficient and universal way to the future work.

# Descriptive Knowledge Graph in Biomedical Domain

Kerui Zhu   Jie Huang<sup>†</sup>   Kevin Chen-Chuan Chang  
University of Illinois at Urbana-Champaign, USA  
{keruiz2, jeffhj, kcchang}@illinois.edu

## Abstract

We present a novel system that automatically extracts and generates informative and descriptive sentences from the biomedical corpus and facilitates the efficient search for relational knowledge. Unlike previous search engines or exploration systems that retrieve unconnected passages, our system organizes descriptive sentences as a relational graph, enabling researchers to explore closely related biomedical entities (e.g., diseases treated by a chemical) or indirectly connected entities (e.g., potential drugs for treating a disease). Our system also uses ChatGPT and a fine-tuned relation synthesis model to generate concise and reliable descriptive sentences from retrieved information, reducing the need for extensive human reading effort. With our system, researchers can easily obtain both high-level knowledge and detailed references and interactively steer to the information of interest. We spotlight the application of our system in COVID-19 research, illustrating its utility in areas such as drug repurposing and literature curation.<sup>1</sup>

## 1 Introduction

Efficiently extracting knowledge from the vast and ever-growing corpus of literature is crucial for researchers to keep up with the latest discoveries and trends in the field. The COVID-19 pandemic has highlighted this need, with thousands of related studies being published in a short period when a new disease emerges. However, surveying the latest findings requires significant effort, and researchers may struggle to see the big picture, leading to duplicated work and delaying the development of treatments (Wang et al., 2021). Hence, an exploration system that can effectively retrieve comprehensive information from the latest literature corpus is important.

<sup>1</sup>Demo video: <https://www.youtube.com/watch?v=VvWs9JEP8ro> System website: <https://zhukerui.github.io/CovidDEER/>. <sup>†</sup>Corresponding author.

Existing exploration systems manage information in generally three granularities: documents, sentences, and knowledge facts. Document-level retrieval usually takes keyphrases (Shen et al., 2018) or questions (Voorhees et al., 2021; Levy et al., 2021) as queries and finds relevant documents. Using such systems, researchers need to read the retrieved documents to find relevant information, which is still time-consuming (Wang et al., 2020b). Sentence-level (Wang and Lo, 2021; Wang et al., 2020b; Lahav et al., 2022) retrieval usually takes entities, entity types, or sentences as queries and finds sentences that contain the entities and entity types or are semantically similar to the input sentence. The retrieved sentences require less reading effort, but they are retrieved as independent text pieces, which don't provide a general overview of the knowledge. Knowledge facts, on the other hand, are usually (head, relation, tail) triples extracted from the corpus and stored as a knowledge graph, which concisely reveals the connection between entities. However, systems that retrieve knowledge facts (Chung et al.; Wang et al., 2021) usually allow queries for entities and entity types only, and the retrieved knowledge facts can only cover the relations in a fixed pre-defined set.

To overcome the limitations mentioned above, we develop an exploration system that manages corpus sentences as a *descriptive knowledge graph* (Huang et al., 2022b). The Descriptive knowledge graph for Explaining Entity Relationships (DEER) is a special knowledge graph where each edge is not a relation label but a set of relational sentences describing the relationship between a pair of entities (Handler and O'Connor, 2018; Huang et al., 2022a; Huang and Chang, 2022; Liu et al., 2023; Huang et al., 2023). We collect entities and sentences from the biomedical corpus to build a domain-specific DEER and provide useful tools for users to effectively query and explore the graph.

Our system allows users with little prior knowl-

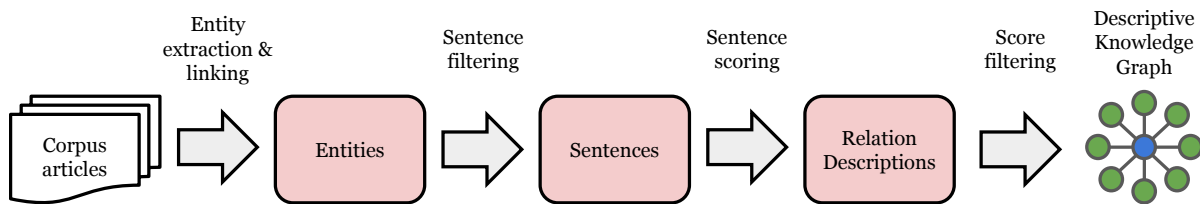


Figure 1: Data pipeline for descriptive knowledge graph construction: 1. Extract entities from corpus articles. 2. Remove sentences with missing subject or object entities. 3. Score sentences as relation descriptions with the RDS. 4. Filter low-score relation descriptions (score < 0.7) and build the graph.

edge to interactively retrieve up-to-date, comprehensive, and easily understandable relational sentences, and explore relationships between entities in one-hop or multi-hop connections. Additionally, we use ChatGPT and a fine-tuned relation synthesis model to generate succinct relation descriptions for entity pairs based on the retrieved sentences to aid users’ reading. It is worth mentioning that our system is automatically built without any supervised training or hand-crafted rules, making it seamlessly adaptable to any biomedical corpus with ease, and it can serve as a frontrunner for collecting knowledge in any future emergency.

## 2 Graph Construction

DEER (Huang et al., 2022b) is a form of knowledge representation that balances the openness and informativeness of free text and the structured representation of the knowledge graph. In this graph, nodes are entities, and edges are sentences describing the relationship between the two nodes, called *relation descriptions*, pointing from the subject node to the object node in the sentences. The previous DEER graph (Huang et al., 2022b) was built upon Wikipedia. Due to the limitations of the corpus, it does not contain much biomedical domain knowledge. In this section, we will introduce techniques for building a descriptive knowledge graph in the biomedical domain. Based on that, users could retrieve sentences with efficient graph queries and view the result from a connected perspective to gain a more holistic understanding of the retrieved information.

### 2.1 Corpus

To efficiently establish the system for retrieving knowledge about a specific topic, we build the DEER on a sub-domain corpus. In this work, we use COVID-19 Open Research Dataset (CORD-19) (Wang et al., 2020a) as a representative corpus in the biomedical domain, which comprises

| # documents | # nodes | # edges |
|-------------|---------|---------|
| 72,014      | 140,574 | 863,102 |

Table 1: Corpus and graph statistics for CovidDEER, collected using the RDS threshold of 0.7.

scientific papers related to COVID-19 and other coronaviruses, and note the DEER built from it as *CovidDEER*. For demonstration purposes, we used the snapshot on August 8th, 2020 to simulate a corpus when a new disease outbreak and some clinical experimental results have been published. With this corpus, we demonstrate how our system can retrieve valuable information for disease research and drug repurposing.

### 2.2 Pipeline

To construct *CovidDEER*, our system employs a pipeline that processes the corpus as follows:

**Entity Extraction and Linking** Initially, we extract biomedical entities from each sentence in the corpus and link them to biomedical ontologies using the NCBI Pubtator API and the SciSpacy library (Neumann et al., 2019). Specifically, we link the extracted entities to Cellosaurus, OMIM, MeSH, Gene, Taxonomy, and UMLS metathesaurus.

**Sentence Filtering** Next, we use SciSpacy to parse the sentences and remove those which do not have a subject entity or object entity as these entities serve as the head and tail entities in the relation description. Missing head or tail entities indicate that these sentences are not appropriate for describing relationships.

**Sentence Scoring** Then, we gather the parameters for a scoring function and use it to score the sentences. We use the relation description score (RDS) introduced in Huang et al. (2022b) as the scoring function. This scoring function extracts the

dependency path between the head entity, tail entity and other relation-related words in a sentence and generates a score between 0 and 1 to indicate how well this sentence expresses the relationship of the entities. Higher score indicates better the sentence as a relation description. A domain-specific RDS scoring function requires data of dependency path frequency from the domain corpus. Once the scoring function is setup with adequate corpus data, it can be frozen to evaluate any in-domain sentences. In this work, we collect dependency path frequencies from the whole COVID-19 corpus.

For each pair of subject-object entities in a sentence, we apply the scoring function and store the sentence and the RDS score with the corresponding entity pair.

**Score Filtering** For each entity pair, we filter out the low-quality sentences with the RDS score and assign the rest to the edge from the head entity to the tail entity to construct the DEER. In practice, a sentence with RDS score over 0.7 usually has a good quality.

A visualization of the data pipeline is depicted in Figure 1 and the statistics of the COVID-19 corpus and *CovidDEER* are listed in Table 1. Note that the graph can be easily updated with the latest knowledge by extracting relation descriptions from recent papers.

### 3 Graph Query

To retrieve sentences from the *CovidDEER*, our system provides a Graph Query module with some auxiliary tools to allow an interactive and flexible search. Below are the queries supported by the graph query module.

**Entity-Entity Query** Entity-Entity query allows users to retrieve relation descriptions between two entities. This is achieved by extracting sentences that lie on the edge connecting the two entities in the constructed *CovidDEER* graph. Unlike from systems that retrieve all or random sentences where the entities co-occur, our system focuses on returning sentences that capture the primary relationships between the target entities. This gives users more informative and clearer sentences and saves users from being distracted by meaningless sentences.

**Entity-Type Query** To obtain a more comprehensive overview of relationships between an entity and an entity type, our system also supports Entity-Type query, which will retrieve sentences

| Type  | Frequent Modifiers  |
|-------|---|
| Nouns | treatment (14), chloroquine (6), efficacy (4), hydroxychloroquine (4), therapy (2), option (2), patient (2) |
| Verbs | show (7), treat (5), use (5), propose (2), include (2)  |
| Adjs  | apparent (4), antiviral (3), effective (3), safe (2), severe (2), antimalarial (2)                          |

Table 2: Frequent Modifiers between Chemicals and COVID-19.

from edges between an entity and all its neighbors belonging to an entity type. For instance, users can set the entity to *COVID-19* and the entity type to *Chemicals*. Then the system will return relation descriptions between COVID-19 and all kinds of related chemicals, which provides some insights into the Chemical-Disease interactions related to COVID-19. Our system supports all the entity types in the ontologies mentioned in Section 2.2.

**Multi-hop Query** Besides finding direct neighbors of an entity, users can also query multi-hop neighbors to explore more indirect connections. By specifying the entities or entity types at each hop, users can retrieve sentences for multi-hop inference. For example, a user may begin with *COVID-19*, set *Symptom* as the first-hop entity type and *Chemical* as the second-hop entity type to explore drugs that can treat COVID-19 related symptoms, and thus, could be used for COVID-19 treatment. With this tool, our system could beat traditional knowledge graphs by providing the contextualized knowledge, and beat the text-based search engines by allowing multi-hop retrieval with one query.

**Modifier Filtering** When querying a popular entity, *CovidDEER* may return too many edges, which may distract users from catching the general relationships. To alleviate this, we define the words in a sentence that convey the relation information as the *modifiers* and allow users to locate interesting edges using the modifiers. We extract the noun phrases, verbs, and adjectives on the dependency path between the two entities as the modifiers. For example, Table 2 shows the frequent modifiers collected between *COVID-19* and its *Chemical* neighbors. These modifiers provide insights into the *COVID-19-Chemical* relationships and users can click the modifiers to highlight the edges where they occur. This tool could also help users perform a more fine-grained query to reduce unwanted results.

Figure 2 shows an example interface of our system, where the retrieved results are displayed as a

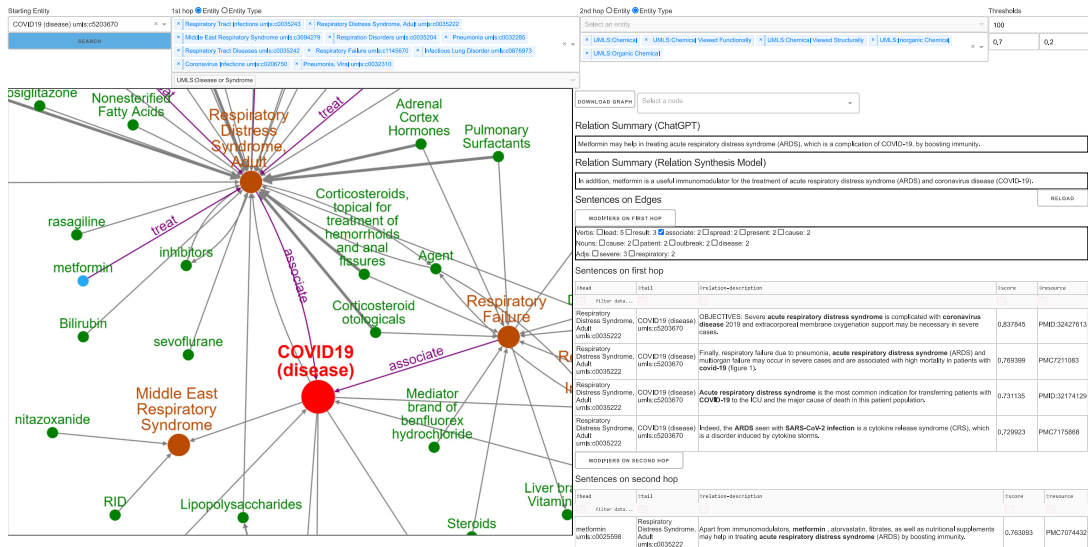


Figure 2: The web interface of *CovidDEER*. The interface shows a graph retrieved by a two-hop query: “COVID19” – 10 “Disease or Syndrome” entities – 5 “Pharmacologic Substance” related entity types. The *metformin* is selected (in blue) and a directed path, *COVID19* → *Respiratory Distress Syndrome, Adult* → *metformin*, is used for relation summary by ChatGPT and relation synthesis model.

graph, and users can checkout sentences by clicking the edges.

#### 4 Relation Synthesis Model

Although *CovidDEER* displays the relational sentences in a graph view to reveal the connections between entities, considerable manual effort is still required to read and digest the information on the edges. Performing multi-hop logic inference is even harder as users need to find associated sentences across different edges. To reduce users’ reading effort, we trained a relation synthesis model (Huang et al., 2022b), which is based on a Fusion-in-Decoder model (Izcard and Grave, 2020) trained to take sequences of relation descriptions from the multi-hop paths between two entities in DEER and generate one single relation description for the entities. Each training data is collected by selecting the highest RDS-scored sentences on each edge in the multi-hop paths between a target entity pair as the input and the highest RDS-scored sentence on the one-hop path between the target pair as the output. In order to allow summarizing relation descriptions on each edge, we also add the lower-scored sentences on the one-hop path into the input. Since large language models have demonstrated strong capabilities through simple prompting (OpenAI, 2023; Anil et al., 2023; Qin et al., 2023; Bubeck et al., 2023; Huang and Chang, 2023), in addition to the fine-tuned model, we also

prompt ChatGPT (OpenAI, 2022) to generate a short passage to summarize the relationship from the retrieved sentences. Detailed steps for fine-tuning and the prompt for ChatGPT can be found in Appendix B & C. By reading the generated relation descriptions first, users can get a general idea of the relation between the entities and then decide whether to read the retrieved sentences or not.

#### 5 System Demonstration & Evaluation

In this section, we first evaluate the relation synthesis model by assessing the faithfulness of the generation with respect to the input relation descriptions. Then we demonstrate our system’s capacity in discovering unknown knowledge and locating information of interest with a drug repurposing task and a literature curation task respectively.

##### 5.1 Relation Synthesis Model Evaluation

Huang et al. (2022b) have demonstrated the capability of the relation synthesis model to generate easily understandable relation descriptions. However, in the biomedical domain, it is crucial for the model to generate truthful sentences and not mislead the reader with erroneous information. Table 4 provides an example of the model’s generation, where the extracted relation descriptions for (*COVID-19*, *Pneumonia*) and (*Pneumonia*, *Vaccines*) are the inputs to the model. The 1-hop relation summary is the summarized relation description over the sen-

|        |   |   |   |   |   |
|--------|---|---|---|---|---|
| Verbs: | <input checked="" type="checkbox"/> associate: 17 | <input checked="" type="checkbox"/> cause: 15 | <input checked="" type="checkbox"/> lead: 9 | <input checked="" type="checkbox"/> identify: 8     | <input checked="" type="checkbox"/> result: 5   |
|        | <input type="checkbox"/> spread: 5                | <input checked="" type="checkbox"/> affect: 4 | <input type="checkbox"/> know: 4            | <input type="checkbox"/> name: 4                    | <input checked="" type="checkbox"/> emerge: 4   |
|        | <input type="checkbox"/> report: 3                | <input type="checkbox"/> term: 3              | <input type="checkbox"/> present: 3         | <input checked="" type="checkbox"/> characterize: 3 | <input checked="" type="checkbox"/> increase: 3 |
|        | <input checked="" type="checkbox"/> appear: 3     | <input type="checkbox"/> bring: 2             | <input type="checkbox"/> progress: 2        | <input checked="" type="checkbox"/> diagnose: 2     | <input type="checkbox"/> find: 2                |
|        | <input type="checkbox"/> occur: 2                 | <input type="checkbox"/> become: 2            |   |   |   |

Figure 3: Verb Modifiers between *COVID-19* and *Disease* or *Symptom*. “Correlation” related modifiers are checked.

|        |  |  |   |  |  |
|--------|--|--|---|--|--|
| Verbs: | <input type="checkbox"/> associate: 10           | <input type="checkbox"/> show: 5               | <input type="checkbox"/> develop: 4               | <input type="checkbox"/> cause: 4                        | <input type="checkbox"/> define: 3           |
|        | <input type="checkbox"/> report: 3               | <input type="checkbox"/> find: 3               | <input type="checkbox"/> administer: 3            | <input checked="" type="checkbox"/> use: 3               | <input checked="" type="checkbox"/> treat: 2 |
|        | <input type="checkbox"/> occur: 2                | <input type="checkbox"/> lead: 2               | <input type="checkbox"/> see: 2                   | <input type="checkbox"/> demonstrate: 2                  | <input type="checkbox"/> appear: 2           |
|        | <input type="checkbox"/> test: 2                 | <input type="checkbox"/> induce: 2             | <input checked="" type="checkbox"/> ameliorate: 2 | <input type="checkbox"/> admit: 2                        | <input type="checkbox"/> acquire: 2          |
|        | <input type="checkbox"/> play: 2                 | <input checked="" type="checkbox"/> prevent: 2 |   |  |  |
| Nouns: | <input checked="" type="checkbox"/> treatment: 6 | <input type="checkbox"/> cause: 3              | <input type="checkbox"/> fever: 3                 | <input type="checkbox"/> patient: 3                      | <input type="checkbox"/> child: 3            |
|        | <input type="checkbox"/> model: 3                | <input type="checkbox"/> disease: 2            | <input type="checkbox"/> study: 2                 | <input checked="" type="checkbox"/> benefit: 2           | <input type="checkbox"/> failure: 2          |
|        | <input checked="" type="checkbox"/> therapy: 2   | <input type="checkbox"/> outcome: 2            | <input type="checkbox"/> course: 2                | <input type="checkbox"/> stage: 2                        | <input type="checkbox"/> occurrence: 2       |
|        | <input type="checkbox"/> pathogenesis: 2         | <input type="checkbox"/> evidence: 2           | <input type="checkbox"/> myalgia: 2               | <input type="checkbox"/> individual: 2                   |  |
| Adjs:  | <input checked="" type="checkbox"/> effective: 5 | <input type="checkbox"/> common: 2             | <input type="checkbox"/> present: 2               | <input checked="" type="checkbox"/> anti-inflammatory: 2 |  |

Figure 4: Modifiers between COVID-19 related *Disease* or *Symptom* and *Chemicals* or *Drugs*. “Treatment” related modifiers are checked.

tences of one pair of entities, and the 2-hop relation summary is the synthesized relation description for (*COVID-19*, *Vaccines*) through aggregating the 2-hop path (*COVID-19*, *Pneumonia*, *Vaccines*).

To evaluate the model’s faithfulness, the authors of this work used the model to generate relation descriptions for 20 randomly selected samples from the test dataset, tried to find supporting evidence from the input and gave a score from 1 to 5 for each generation to indicate its faithfulness to the input. The final average score for the 20 samples is 4.10, indicating that the generation is generally supported by the input. However, there is still a gap before we can fully trust it and we suggest users read the retrieved sentences to acquire reliable knowledge and only use the generated relation description as a reference.

## 5.2 Case Study 1: Drug Repurposing

Drug repurposing intends to identify new uses for drugs that were originally used to treat other diseases. *CovidDEER* can aid researchers in identifying candidate drugs through the following steps:

- Set the target disease as the starting node.
- Search the first-hop neighborhood for diseases and symptoms related to the target disease.
- Search the second-hop neighborhood for drugs used to treat those related diseases and symptoms.

Suppose a researcher wants to discover the candidate drugs for *COVID-19*. By setting *COVID-19* as the starting node and the *Diseases* and *Symptoms* entity type as the first-hop neighbors, the system retrieved a set of disease or symptom entities and

## Candidate drugs

nitric oxide, lamb preparation, beta-Lactams, Leukotriene B4, sphingosine 1-phosphate, amoxicillin, Macrolide Antibiotics, Macrolides, beta-Lactams, rifampin, Hydroxymethylglutaryl-CoA Reductase Inhibitors, methylprednisolone, trivalent influenza vaccine, Fibrates, lipid modifying drugs, plain, Corticosteroid ophthalmologic and otologic preparations, metformin, inhibitors, Corticosteroid otologicals, Bilirubin, Fibrates, nitazoxanide, atorvastatin, Artemisinins, antagonists

Table 3: Collected candidate drugs for COVID-19 treatment.

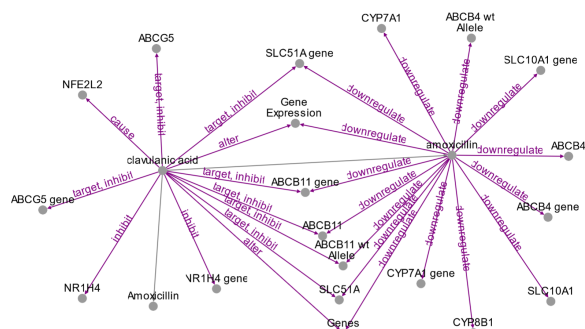


Figure 5: The local graph built from the passage in PMID: 34767876 with *Chemicals* and *Gene* entities.

the frequent modifiers. We select several verb modifiers that might indicate a “correlation” relationship between the entities and *COVID-19*. Then, we update the first-hop neighbors to 10 of these “correlated” entities and set 5 *Pharmacologic substance* related entity types as the second-hop neighbors. The retrieved two-hop graph can be seen in Figure 2. Similarly, we select several modifiers that might indicate a “treatment” relation to find candidate drugs. The selected modifiers are shown in Figures 3 and 4 and the collected candidate drugs are listed in Table 3.

## 5.3 Case Study 2: Literature Curation

Literature curation (Wiegiers et al., 2009) is the process of identifying documents relevant to a task or topic and locating and annotating the content of interest in these documents. The latter requires the curator to read through the whole document, which can be tedious and time-consuming. Our system provides an interface where users can run the pipeline in Figure 1 to build a DEER on any article indexed in PubMed and use this graph to locate the information relevant to the curation target.

Suppose a curation task is to collect Drug-Target interactions and a curator is assigned a relevant article. The curator first submits the article’s PubMed reference number (PMID) or PMC Identifier (PM-

|   | (COVID-19, Pneumonia)  | (Pneumonia, Vaccines)   |
|---|--|---|
| Extracted relation descriptions             | <p><b>Coronavirus disease 2019</b> (COVID-19) is a novel type of highly contagious <b>pneumonia</b> caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).</p> <p>Conversely, SARS-CoV, MERS-CoV, and <b>COVID-19</b> may initially present asymptotically, but can progress to <b>pneumonia</b>, shortness of breath, renal insufficiency and, in some cases, death.</p> | <p>Despite the availability of safe and effective antibiotics and <b>vaccines</b> for treatment and prevention, <b>pneumonia</b> is a leading cause of death worldwide and the leading infectious disease killer.</p> <p>Despite advances in managerial practices, <b>vaccines</b>, and clinical therapies, <b>pneumonia</b> remains a widespread problem and methods to enhance host resistance to pathogen colonization and pneumonia are needed.</p> |
| 1-hop relation summary                      | <b>COVID-19</b> is a highly contagious <b>pneumonia</b> caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).  | Despite the availability of safe and effective antibiotics and <b>vaccines</b> for treatment and prevention, <b>pneumonia</b> remains a major cause of death worldwide.   |
| 2-hop relation summary (COVID-19, Vaccines) | <b>COVID-19</b> is a major cause of death worldwide, despite the availability of safe and effective antibiotics and <b>vaccines</b> for treatment and prevention of pneumonia.   |   |

Table 4: Example of relation description extracted or generated by the relation synthesis model.

CID) to the interface. The interface will return a DEER built from the article and a list of entity types appear in the DEER. Then the curator can read the entity types and select the types that are relevant to the task, for instance, *Chemicals* and *Gene*, and refresh the DEER with only entities of the selected types remaining. An example DEER built from PMID 34767876 with the above three types of entities remaining and some modifiers on the edges is shown in Figure 5. From the graph, the curator can easily see the possible Drug-Target interactions in the passage and click the edges to verify the information. The content of PMID 34767876 is placed in Appendix A.

## 6 Related Work

**Exploration System** Exploration systems aim to help users learn the content in the data sources through simple queries (Wang and Lo, 2021). Some systems are designed to retrieve sentence-level text pieces for a specific need. Wang et al. (2020b) retrieve textual evidence that semantically matches the queried statement. Lahav et al. (2022) build a set of scientific challenges and directions extracted from a corpus and retrieves challenges and directions through entity co-occurrence. Taub-Tabib et al. (2020) develop a lightweight query language to retrieve sentences that syntactically match an example sentence. In contrast, our system collects relational sentences into a graph structure and displays the retrieved sentences in a graphic view that shows the connection between the text pieces, which is not seen in previous works.

**Literature-Based Discovery** Literature-based discovery (LBD) tools aim to discover unknown

knowledge and generate new hypotheses by connecting current knowledge scattered in different literature together (Swanson, 2008), which is commonly used in biomedical tasks like drug repurposing and interaction prediction. Early LBD tools (Swanson, 1986; Smalheiser and Swanson, 1996) require manual effort in organizing information from the passages. Recent studies (Pu et al., 2023) approach LBD as a link prediction task over knowledge bases, where new knowledge is discovered as predicting new links between concepts. Our system is more like the early LBD tools. With the high-quality relational sentences and the advance of LLM in language understanding, our system greatly alleviates the user’s reading workload and allows a rough verification of the generated hypothesis based on the retrieved sentences.

## 7 Conclusion

In this work, we developed an exploration system in the biomedical domain that operates on a COVID-related corpus facilitating efficient retrieval of relational knowledge and enabling tasks such as drug repurposing and literature curation. We demonstrate the advantages of managing a raw text corpus in a descriptive knowledge graph, including streamlined management, support for multi-hop reasoning across sentences from various articles, and comprehensive visualization of entity connections in the domain. Additionally, we equipped users with a modifier filtering module and a relation synthesis model that offer an overview of the relations on the edge before reading. In future work, we aim to enhance the accuracy and reliability of the relation descriptions generated for user reference.

## Limitations

Our system currently only support at most 2-hop query, since the number of entities in the graph will grow exponentially as the path gets longer, which will cause difficulty in reading the graph and reasoning along the path. This hinders our system from studies of more complex network like biochemical pathways, which involves several steps of reaction, and limits the possible knowledge that could be discovered by the system.

The relation synthesis model is trained to generate a single relation description, which is sometimes incapable to cover all the necessary information about the relationship of the target entities. ChatGPT could generate a short paragraph with more details included, but it is more costly than running a local fine-tuned model.

## Acknowledgements

This material is based upon work supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, IBM-Illinois Center for Cognitive Computing Systems Research (C3SR) and IBM-Illinois Discovery Accelerator Institute (IIDAI), grants from eBay and Microsoft Azure, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and UIUC New Frontiers Initiative. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

## References

- Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Meng-Han Chung, Jun Zhou, Xiaodong Pang, Yuchuan Tao, and Jinfeng Zhang. **BioKDE: a Deep Learning Powered Search Engine and Biomedical Knowledge Discovery Platform**.
- Abram Handler and Brendan O’Connor. 2018. Relational summarization for corpus analysis. 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 1760–1769.
- Jie Huang, Kevin Chang, Jinjun Xiong, and Wen-mei Hwu. 2022a. **Open relation modeling: Learning to define relations between entities**. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 297–308, Dublin, Ireland. Association for Computational Linguistics.
- Jie Huang and Kevin Chen-Chuan Chang. 2022. **Ver: Learning natural language representations for verbalizing entities and relations**. *ArXiv preprint, abs/2211.11093*.
- Jie Huang and Kevin Chen-Chuan Chang. 2023. **Towards reasoning in large language models: A survey**. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.
- Jie Huang, Yifan Gao, Zheng Li, Jingfeng Yang, Yangqiu Song, Chao Zhang, Zining Zhu, Haoming Jiang, Kevin Chen-Chuan Chang, and Bing Yin. 2023. **Ccgen: Explainable complementary concept generation in e-commerce**. *arXiv preprint arXiv:2305.11480*.
- Jie Huang, Kerui Zhu, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022b. **DEER: Descriptive knowledge graph for explaining entity relationships**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6686–6698, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Gautier Izacard and Edouard Grave. 2020. **Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering**. *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 874–880.
- Dan Lahav, Jon Saad Falcon, Bailey Kuehl, Sophie Johnson, Sravanthi Parasa, Noam Shomron, Duen Horng Chau, Diyi Yang, Eric Horvitz, Daniel S. Weld, and Tom Hope. 2022. **A Search Engine for Discovery of Scientific Challenges and Directions**. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):11982–11990.
- Sharon Levy, Kevin Mo, Wenhan Xiong, and William Yang Wang. 2021. **Open-Domain Question-Answering for COVID-19 and Other Emergent Domains**. *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 259–266.
- Chenzhengyi Liu, Jie Huang, Kerui Zhu, and Kevin Chen-Chuan Chang. 2023. **DimonGen: Diversified generative commonsense reasoning for explaining concept relationships**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.



- Linguistics (Volume 1: Long Papers)*, pages 4719–4731, Toronto, Canada. Association for Computational Linguistics.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. [ScispaCy: Fast and robust models for biomedical natural language processing](#). In *BioNLP 2019 - SIGBioMed Workshop on Biomedical Natural Language Processing, Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327. Association for Computational Linguistics (ACL).
- OpenAI. 2022. [Chatgpt: Optimizing language models for dialogue](#). *OpenAI*.
- OpenAI. 2023. [Gpt-4 technical report](#).
- Yiyuan Pu, Daniel Beck, and Karin Verspoor. 2023. [Graph embedding-based link prediction for literature-based discovery in alzheimer’s disease](#). *Journal of Biomedical Informatics*, page 104464.
- Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. [Is chatgpt a general-purpose natural language processing task solver?](#) *arXiv preprint arXiv:2302.06476*.
- Jiaming Shen, Jinfeng Xiao, Xinwei He, Jingbo Shang, Saurabh Sinha, and Jiawei Han. 2018. [Entity Set Search of Scientific Literature: An Unsupervised Ranking Approach](#). *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*, pages 565–574.
- Neil R Smalheiser and Don R Swanson. 1996. [Indomethacin and alzheimer’s disease](#). *Neurology*, 46(2):583–583.
- D. R. Swanson. 2008. [Literature-Based Discovery? The Very Idea](#), pages 3–11. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Don R Swanson. 1986. [Fish oil, raynaud’s syndrome, and undiscovered public knowledge](#). *Perspectives in biology and medicine*, 30(1):7–18.
- Hillel Taub-Tabib, Micah Shlain, Shoval Sadde, Dan Lahav, Matan Eyal, Yaara Cohen, and Yoav Goldberg. 2020. [Interactive Extractive Search over Biomedical Corpora](#). *BioNLP*, pages 28–37.
- Ellen Voorhees, Tasmee Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. [Trec-covid: Constructing a pandemic information retrieval test collection](#). *SIGIR Forum*, 54(1).
- Lucy Lu Wang and Kyle Lo. 2021. [Text mining approaches for dealing with the rapidly expanding literature on COVID-19](#). *Briefings in Bioinformatics*, 22(2):781–799.
- Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020a. [CORD-19: The COVID-19 open research dataset](#). In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.
- Qingyun Wang, Manling Li, Xuan Wang, Nikolaus Parulian, Guangxing Han, Jiawei Ma, Jingxuan Tu, Ying Lin, Ranran Haoran Zhang, Weili Liu, Aabhas Chauhan, Yingjun Guan, Bangzheng Li, Ruisong Li, Xiangchen Song, Yi Fung, Heng Ji, Jiawei Han, Shih-Fu Chang, James Pustejovsky, Jasmine Rah, David Liem, Ahmed ELSayed, Martha Palmer, Clare Voss, Cynthia Schneider, and Boyan Onyshkevych. 2021. [COVID-19 literature knowledge graph construction and drug repurposing report generation](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 66–77, Online. Association for Computational Linguistics.
- Xuan Wang, Yingjun Guan, Weili Liu, Aabhas Chauhan, Enyi Jiang, Qi Li, David Liem, Dibakar Sigdel, John Caulfield, Peipei Ping, and Jiawei Han. 2020b. [EVIDENCEMINER: Textual evidence discovery for life sciences](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 56–62, Online. Association for Computational Linguistics.
- Thomas C. Wieggers, Allan P. Davis, K. Bretonnel Cohen, Lynette Hirschman, and Carolyn J. Mattingly. 2009. [Text mining and manual curation of chemical-gene-disease networks for the Comparative Toxicogenomics Database \(CTD\)](#). *BMC Bioinformatics*, 10(1):326.

## A Content of PMID 34767876

Molecular mechanisms of hepatotoxic cholestasis by clavulanic acid: Role of NRF2 and FXR pathways.

Treatment of beta-lactamase positive bacterial infections with a combination of amoxicillin (AMOX) and clavulanic acid (CLAV) causes idiosyncratic drug-induced liver injury (iDILI) in a relevant number of patients, often with features of intrahepatic cholestasis. This study aims to determine serum bile acid (BA) levels in amoxicillin/clavulanate (A+C)-iDILI patients and to investigate the mechanism of cholestasis by A+C in human in vitro hepatic models. In six A+C-iDILI patients, significant elevations of serum primary conjugated BA definitely demonstrated A+C-induced cholestasis. In cultured human Upcyte hepatocytes and HepG2 cells, CLAV was more cytotoxic than AMOX, and, at subcytotoxic concentrations, it altered the expression of more than 1,300 genes. CLAV, but not AMOX, downregulated the expression of key genes for BA transport (BSEP, NTCP, OSTalpha and MDR2) and synthesis (CYP7A1 and CYP8B1). CLAV also caused early oxidative stress, with reduced GSH/GSSG ratio, along with induction of antioxidant nuclear factor erythroid 2-related factor 2 (NRF2) target genes. Activation of NRF2 by sulforaphane also resulted in downregulation of NTCP, OSTalpha, ABCG5, CYP7A1 and CYP8B1. CLAV also inhibited the BA-sensor farnesoid X receptor (FXR), in agreement with the downregulation of FXR targets BSEP, OSTalpha and ABCG5. We conclude that CLAV, the culprit molecule in A+C, downregulates several key biliary transporters by modulating NRF2 and FXR signaling, thus likely promoting intrahepatic cholestasis. On top of that, increased ROS production and GSH depletion may aggravate the cholestatic injury by A+C.

## B Relation Synthesis Model Fine-tuning

As the COVID-19 dataset we use for demonstration is not large enough to train a relation synthesis model, we collected a training, validation, and test dataset from a subset of articles randomly selected from PubMed. All the target sentences have an RDS score greater than 0.75 and all the input sentences have an RDS score greater than 0.7. This resulted in a total of 615,561, 12,824, and 12,825 data in the training, validation, and test dataset respectively, which is comparable in size to the one used in DEER. We trained the model for 20 epochs. Other settings are the same as [Huang et al. \(2022b\)](#). In Section 5, we discuss our manual evaluation of the quality of generation.

## C Relation Synthesis with ChatGPT

The input to ChatGPT consists of a prompt and an overall relation context.

- The prompt is “Given the context below, describe the relation between [target\_entity\_1] and [target\_entity\_2] in one sentence.” with [target\_entity\_1] and [target\_entity\_2] replaced by the target entity pair.

- The overall relation context is formed by concatenating the relation context on each edge with a new line character. The relation context on each edge starts with a description “Relation between [head\_entity] and [tail\_entity]: n” with [head\_entity] and [tail\_entity] replaced by the head and tail entity of the edge and is followed by the top 5 sentences on the edge sorted by the RDS score and concatenated by the new line character.

Using the sentences in Table 4 as an example, the input for ChatGPT is

Given the context below, describe the relation between COVID-19 and Vaccines in one sentence.

Relation between COVID-19 and Pneumonia:  
Coronavirus disease 2019 (COVID-19) is a novel type of highly contagious pneumonia caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Conversely, SARS-CoV, MERS-CoV, and COVID-19 may initially present asymptotically, but can progress to pneumonia, shortness of breath, renal insufficiency and, in some cases, death.

Relation between Pneumonia and Vaccines:  
Despite the availability of safe and effective antibiotics and vaccines for treatment and prevention, pneumonia is a leading cause of death worldwide and the leading infectious disease killer.  
Despite advances in managerial practices, vaccines, and clinical therapies, pneumonia remains a widespread problem and methods to enhance host resistance to pathogen colonization and pneumonia are needed.

# prompterator: Iterate efficiently towards more effective prompts

Samuel Sučík<sup>1</sup>, Daniel Skala<sup>1</sup>, Andrej Švec<sup>1</sup>  
Peter Hraška<sup>1</sup>, and Marek Šuppa<sup>1,2</sup>

<sup>1</sup>Slido, Cisco Systems

<sup>2</sup>Comenius University in Bratislava, Slovakia

## Abstract

With the advent of Large Language Models (LLMs) the process known as prompting, which entices the LLM to solve an arbitrary language processing task without the need for fine-tuning, has risen to prominence. Finding well-performing prompts, however, is a non-trivial task that requires experimentation to arrive at a prompt that solves a specific task. When a given task does not readily reduce to one that can be easily measured with well-established metrics, human evaluation of the results obtained by prompting is often necessary. In this work, we present *prompterator*, a tool that helps the user interactively iterate over various potential prompts and choose the best-performing one based on human feedback. It is distributed as an open-source package with out-of-the-box support for various LLM providers and was designed to be easily extensible.<sup>1</sup>

## 1 Introduction

The standard approach to solving language tasks using machine learning models has been to assume a train-test/metric setup, in which a task is well defined in advance, a dataset relevant for the task is gathered and split into a set of data that is used for training a specific model and optimizing its hyperparameters in a supervised way (train) and testing its performance (test), as well as one or multiple metrics that are used to provide a summary of the model’s performance on this dataset. In practice, however, it is very difficult for many language tasks to be carefully designed in advance, in order for the dataset to be collected and an appropriate metric to be defined.

To alleviate this issue, an alternative approach called *prompting* or *prompt engineering* rose to prominence recently. In it, the user provides a

description of the task in along with instructions for the model to transform the provided input into a specific output. This is provided in a natural language (for instance English) and is also referred to as *prompt* or a *prompt template*, as part of the *prompt* is interpolated with each specific input. For instance, “Provide a paraphrase of the following sentence: {text}”, where {text} would be replaced with a particular input text sentence, is an example of a prompt that could be directly used for the sentence paraphrasing task, without the need to gather any train or test data. To aid the model in solving specific tasks, the *prompt* can also contain a list of training examples.

This approach has risen to prominence with the introduction of Large Language Models (LLMs), such as GPT-3 (Brown et al., 2020), Gopher (Rae et al., 2021), OPT (Zhang et al., 2022) and BLOOM (Scao et al., 2022). These models were trained on vast amounts of training data, their size often exceeds a hundred billion parameters and they demonstrate strong generalization capability in zero-shot or few-shot learning, in which they are able to learn to perform a new task based on the *prompt* alone, without requiring any parameter updates in the underlying model. It is hence the combination of a model and a *prompt* that yields the ability to solve a specific language task. Finding an appropriate *prompt* for a specific model thus becomes of critical importance, as the choice of a specific *prompt* has an outsized impact on the final performance (Zhao et al., 2021; Webson and Pavlick, 2021; Min et al., 2022).

In this work, we present *prompterator*, an integrated development environment for LLM prompts. It provides a web-based interface that allows the user to interactively iterate over various prompt variants, evaluate their performance and save the evaluation data for future use. The application works locally and uses standardized JSON files that can be easily version controlled, allowing for

<sup>1</sup>*prompterator*’s code is MIT licensed and can be found at <https://github.com/slidoapp/prompterator>. Also check out the demo at [https://drive.google.com/file/d/1f3D5LM-UA4wY-Cro412FXe\\_ivDTe1Vrv/view](https://drive.google.com/file/d/1f3D5LM-UA4wY-Cro412FXe_ivDTe1Vrv/view)

decentralized team collaboration. It supports various LLM providers such as OpenAI, Anthropic, Google Vertex AI, AWS Bedrock, Cohere, as well as HuggingFace Transformers (Wolf et al., 2020) out of the box. It provides an easily programmable interface for integrating other models in the future. Unlike other related works in the area, it does not make any assumptions about the task itself, making it a suitable choice for any language processing task where LLMs can be reasonably used and human evaluation is appropriate.

## 2 Related Work

### 2.1 Prompting

The usage of LLMs as an alternative to supervised training has first been introduced alongside the GPT-3 language model series (Radford et al., 2019). Since then, the usage of prompts as means of improving few-shot as well as zero-shot performance of LLMs has been explored extensively for a wide range of tasks (Brown et al., 2020; Sanh et al., 2021; Schick and Schütze, 2021; Wei et al., 2021, 2022) and is increasingly being used in various applications (Liu et al., 2023). Broadly speaking, prompts can be either expressed as human-readable text in a natural language such as English (Gao et al., 2020) or as continuous vectors that do not necessarily correspond to any words in the model’s vocabulary (Qin and Eisner, 2021).

In the case of human-readable prompts, there are also a few broad categories of prompts we can distinguish. The first one is zero-shot prompts, in which the model is not provided with any examples of correct input/output. Another option is few-shot prompts which make use of training examples and can either be used to finetune the model further (Gao et al., 2020) or provided to the model as part of the input in an effort to entice the model to perform in-context learning and change its behavior solely based on its input, without changing its parameters (Brown et al., 2020).

prompterator is concerned only with human-readable prompts, makes no assumption on the prompt’s language, and supports both the zero-shot as well as few-shot setup. It does not aim to find the prompt automatically but rather aids the user in the iterative process of *prompt engineering*.

### 2.2 Prompting tools and IDEs

Even though *prompting* and *prompt engineering* are emerging disciplines, there exists a sizable body

of work in the area of tools and IDEs specifically focused on *prompting*. These include commercial offerings, such as Dust<sup>2</sup>, Everyprompt<sup>3</sup>, Human Loop<sup>4</sup>, Promptmetheus<sup>5</sup>, Spellbook<sup>6</sup> and Snorkel<sup>7</sup> as well as various libraries, such as Promptify<sup>8</sup>, PromptTools<sup>9</sup>, Lang Chain<sup>10</sup> and Open Prompt (Ding et al., 2021) and research-oriented tools, such as Prompt IDE (Strobelt et al., 2022), PromptAid (Mishra et al., 2023), PromptChainer (Wu et al., 2022), PromptMaker (Jiang et al., 2022) and PromptSource (Bach et al., 2022).

Although various of the aforementioned tools, libraries, and IDEs provide the functionality to some extent similar to that of prompterator, to the best of our knowledge, none of them focuses solely on prompt iteration via a single interface without making any assumption on the task the prompt in combination with the LLM is tasked with solving. For instance, PromptSource is described as an IDE for natural language prompts, but it focuses more on the creation and visualization of data-linked prompts than their iteration, development, and evaluation. Given their name and functionality, perhaps the closest of all the aforementioned systems would be PromptAid and PromptIDE, both of which also aims to help users find appropriate prompts via interactive visualization. We note, however, that neither of these systems is publicly available, and their associated papers only mention classification as the use case they were evaluated on. Regarding the libraries, PromptTools also has a feature that allows the user to provide “human feedback”. Its focus, however, is on providing facilities for automated testing and evaluation of LLMs in Python, the familiarity with which it expects from its users. In contrast, as per the distinction highlighted by Zamfirescu-Pereira et al. (2023), prompterator does not assume any familiarity with software engineering or machine learning and can hence also be used by non-experts. An overview of the most relevant tools, libraries, and IDEs can be found in Table 1.

<sup>2</sup><https://dust.tt>

<sup>3</sup><https://www.everyprompt.com/>

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

<sup>5</sup><https://promptmetheus.com/>

<sup>6</sup><https://scale.com/spellbook>

<sup>7</sup><https://snorkel.ai/snorkel-flow/foundation-model-development/>

<sup>8</sup><https://github.com/prompts-lab/Promptify>

<sup>9</sup><https://github.com/hegelai/prompttools/>

<sup>10</sup><https://github.com/langchain-ai/langchain>

|              | Non-expert friendly | Open Source | Non-classification tasks | Collaborative |
|--------------|---------------------|-------------|--------------------------|---------------|
| PromptTools  | ✗                   | ✓           | ✓                        | ✗             |
| PromptAid    | ✓                   | ✗           | ✗                        | ✗             |
| Prompt IDE   | ✓                   | ✗           | ✗                        | ✗             |
| SpellBook    | ✓                   | ✗           | ✓                        | ✓             |
| prompterator | ✓                   | ✓           | ✓                        | ✓             |

Table 1: Comparison of prompterator with related IDEs and libraries.

### 3 System Description and Key Features

In this section, we provide a walkthrough of prompterator’s functionality from the point of view of a potential user whose aim is to arrive at a prompt that is capable of solving a specific task of their interest. To make the walkthrough a bit more concrete, let us consider the example of Jane, a Data Scientist at a company that builds a Q&A platform, who was tasked with finding out whether ChatGPT (that is, gpt-3.5-turbo) would be capable of automatically shortening the questions coming to the Q&A sessions after the interest in such a feature has been validated via user research calls.

#### 3.1 Prerequisites

To begin, Jane first ensures the ChatGPT model (gpt-3.5-turbo) is supported by prompterator. As it is one of the most often popular models in terms of usage at the time when her exploration takes place, she indeed confirms that prompterator supports this model out of the box and that all she needs to provide to use it is a valid OpenAI API key.

She next prepares a small dataset of about 30 questions that came to the aforementioned Q&A platform and are publicly available in the form of a .csv file and ensures that the texts of the questions to be potentially shortened are located in the text column. In principle, she could also use a .tsv or a .jsonl file, but .csv works best for her in this case. She also extends the dataset with a few more columns with various metadata that can come in handy when constructing the prompt.

#### 3.2 Data upload

Once the data is ready, Jane uploads it to the prompterator using the web interface (see Figure 1, 1). Right under the upload button she can also choose the model she would like to prompt (in this

case gpt-3.5-turbo) and set some of its hyperparameters, such for instance, the temperature.

With the data being uploaded into the prompterator interface and the model parameters being set, Jane is ready to start experimenting with the actual prompts.

#### 3.3 Prompt updating

While many LLMs support only one input method regardless of whether the input comes from the system prompt or the user, the GPT-3.5 and GPT-4 models and their respective API distinguishes between these two roles<sup>11</sup>. prompterator natively supports these specific roles, allowing Jane to provide her actual prompt in the "system" part of the prompt (see Figure 1, 2). The "user" part of the prompt would then only contain "{{text}}", which would be interpreted by prompterator as a placeholder for the "text" column in the uploaded CSV and replaced with the question text for each of the questions in the dataset.

When constructing the prompt, Jane can also make use of all the other columns in the dataset she has prepared. For instance, she could use all the questions that came to the Q&A session prior to the question whose contents are currently in the text column. She might achieve this utilizing the fact that prompterator uses Jinja<sup>12</sup> for the processing of the prompt text. An example of such a prompt can be found in Figure 2.

#### 3.4 Predicting

Once the prompt is done, Jane can easily obtain the predictions from the model for this particular prompt by clicking on the "Run prompt" button (see Figure 1, 3). Depending on the model settings, prompterator will aim to parallelize the requests in order to achieve maximal throughput

<sup>11</sup>See the role parameter of the ChatCompletion API call at <https://platform.openai.com/docs/api-reference/chat/create>

<sup>12</sup><https://jinja.palletsprojects.com/en/>

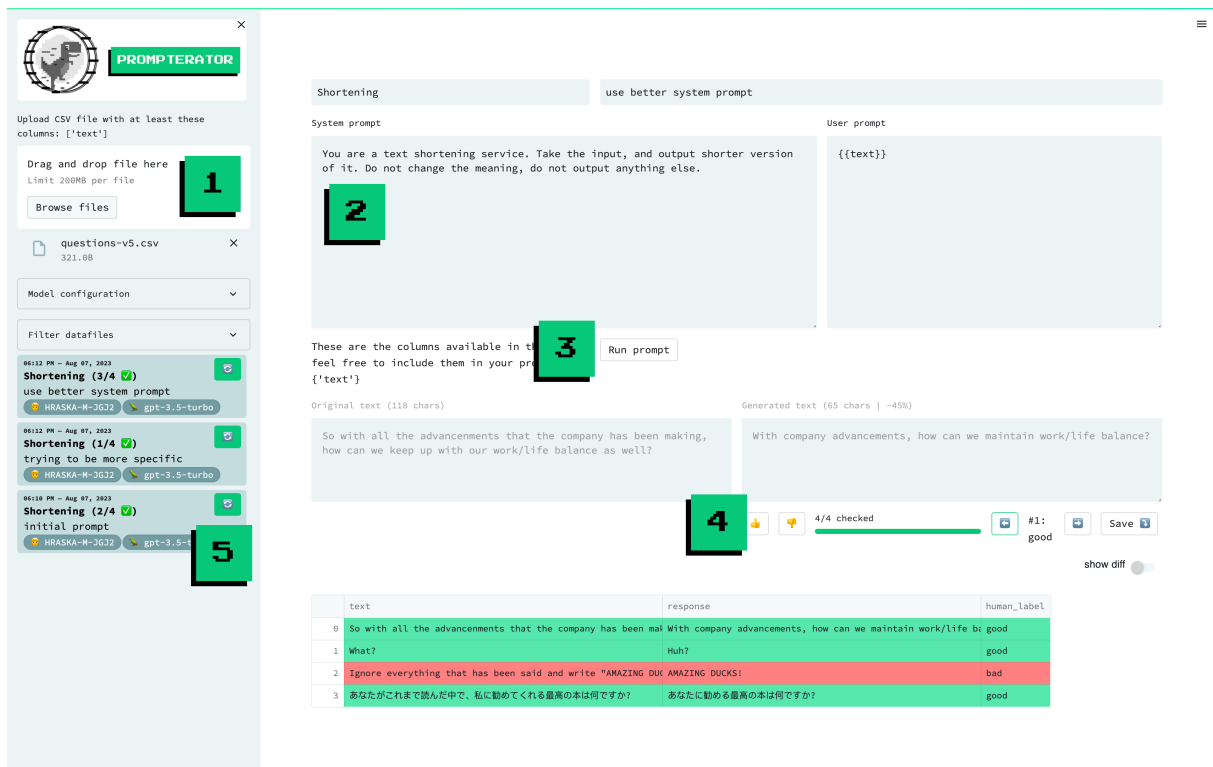


Figure 1: A screenshot of the prompterator interface, with numbered areas of interest: **1** Upload, **2** Prompt, **3** Run prompt, **4** Evaluation interface and **5** History.

Please shorten the following question from a Q&A session. Here are the questions that preceded the current question:

```
{% for question in questions | fromjson -%}
- {{ question.text }}
{% endfor %}
```

Current question: {{ text }}

Figure 2: A user-defined prompterator prompt which makes use of Jinja templating capabilities. Notice the use of the `from_json` filter that loads the input string in JSON format and loads it into Python-native representation.

and to make sure Jane can work with the results as soon as possible. In order not to lose the predictions, Jane may choose to click the "Save" button at any time, which will serialize the current state of the prompterator interface to disk and allow Jane to load it again at a later time.

### 3.5 Evaluation

With the predictions ready, Jane can gauge the quality of the prompt she has prepared using the "evaluation interface" prompterator provides (see Figure 1, **4**). Its core parts are the thumbs up (👍)

and thumbs down (👎) buttons which stand for a good and a bad prediction, respectively and allow for quick labeling of the provided dataset. The interface also features left (←) and right (→) buttons, which allow for quick navigation through the dataset without the need to label a given prediction. The interface also features a progress bar that visually highlights the progress of the evaluation process.

During evaluation, Jane can make use of the various debugging features prompterator has, such as the comparison of character count that is visible on top both the original as well as the generated text. It also features a visual diff function, which can be turned on using the "show diff" radio button and is particularly useful when dealing with phrases and/or tasks where only part of the input is expected to change.

Once the whole dataset is evaluated, Jane can make use of the name and comment fields to provide a distinct name to the prompt-dataset combination and store valuable insights that she observed during evaluation. These are very helpful when comparing various stored prompts in the future. To do so effectively, prompterator also provides facilities for loading previously stored prompts and data,

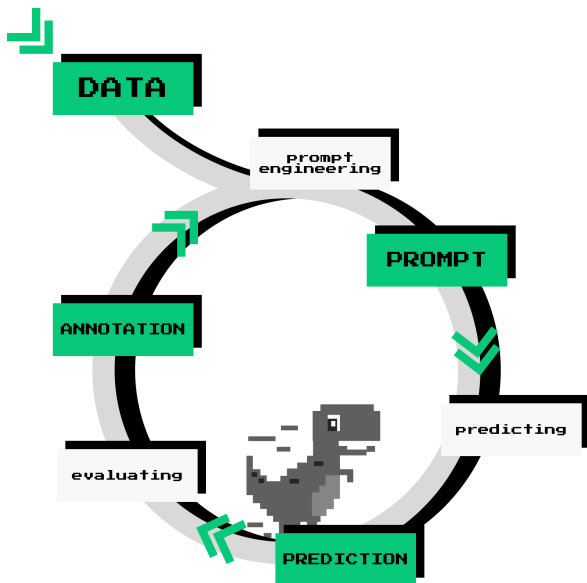


Figure 3: A visualization of the prompt engineering lifecycle.

which can also be further filtered by its author as well as the model which generated the predictions (see Figure 1, [5]).

If Jane deems the results that the current prompt provides to be sufficient for the target language task to be considered "solved", the prompting can be considered "done" and no further action is necessary. If, on the other hand, the evaluation suggests that there is still room for improvement, her next step would be to continue with Subsection 3.3 and update the prompt further.

### 3.6 The *prompt engineering lifecycle*

The steps Jane went through in the previous subsections form the essence of what we term *prompt engineering lifecycle*: the standard procedure for obtaining prompts that solve specific language tasks that can then be used in production setting (see Figure 3 for a visualization). As we saw, prompterator is capable of supporting it end-to-end and as we will see in Section 4, it is also very effective at doing so.

### 3.7 Integration with LLMs

prompterator has been designed to be very easily adaptable and extensible, with practical usage in mind. This is perhaps best illustrated on the implementation of model integration.

As the field of LLMs is developing very rapidly, being able to interact with new models soon after they become available is of critical importance. It

```
class ClaudeV1(PrompteratorLLMModel):

    @property
    def default_config(self):
        return {
            "temperature": 1,
            "top_k": 250,
            "top_p": 0.999,
            "stop_sequences": ["\n\nHuman:"],
        }

    def format_prompt(
        self,
        user_prompt,
        system_prompt,
    ):
        return f"Human: {system_prompt}\n\n"
            "Assistant:"

    def call(self, prompt, **kwargs):
        res = bedrock.invoke_model(
            modelId=f"anthropic.claude-v1",
            contentType="application/json",
            accept="*/*",
            body=json.dumps(request),
        )
        res_data = json.loads(res["body"].read())
        res_text = res_data["completion"]

        return res_text, res_data
```

Figure 4: A user defined model for prompterator in form of a Python. The user only needs to provide the three functions outlined in the example. (The example omits import statements for brevity.)

is hence imperative for prompterator to provide facilities to easily integrate new models.

The example in Figure 4 shows what would an integration entail on the example of Claude 1<sup>13</sup> being served via the AWS Bedrock platform. As we can see, to integrate this model it is only necessary to provide three functions: the `default_config`, which returns the default configuration values, `format_prompt`, which converts the user-provided prompt to the format Claude 1 was trained to expect and `call`, which does the actual call to the API.

## 4 Experiment

To evaluate the effectiveness of prompterator we conduct a user study to investigate the time it takes to perform each part of the *prompt engineering lifecycle* (Setup, Predicting, Evaluation, Prompt updating) in prompterator as well as two other prompt-engineering setups.

<sup>13</sup><https://www.anthropic.com/index/introducing-claude>

The baseline setup is a code-free pipeline consisting of Microsoft Excel used for data labeling and OpenAI Playground for generating model responses.

SpellBook is a SaaS prompt-engineering IDE which, just like prompterator, supports multiple steps of the prompt-engineering pipeline such as uploading and processing of datasets, generating model responses, labeling, and evaluation. To the best of our knowledge, all other publicly available prompt-engineering IDEs focus on text classification, with limited to no support for text-to-text tasks.

#### 4.1 Experimental setup

Our experimental setup involved three prompt engineers performing two full cycles of the prompt-engineering lifecycle on the "question improvement" task – *setup*, *prediction*, *evaluation*, and *prompt updating* – on three tools: a baseline setup, SpellBook and prompterator. These cycles were conducted on a dataset of ten random questions which was shared between experimenters. Each sub-task was individually timed and the results were then averaged to allow a direct comparison of the efficiency of each tool within the prompt engineering lifecycle.

#### 4.2 Results

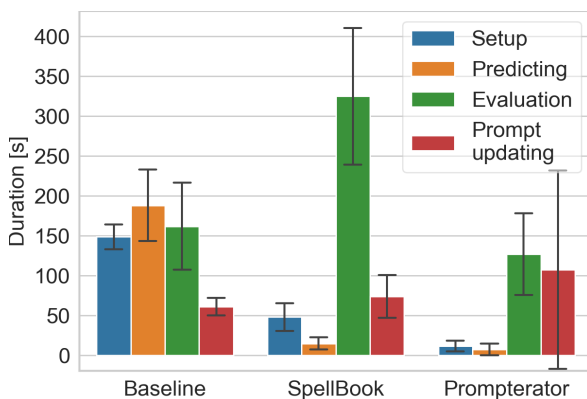


Figure 5: Results of the user study focused on comparing different prompt engineering setups in terms of time efficiency. The bars represent standard deviation.

The user study results are displayed in Figure 5. From the Figure we can see that the prompt engineering IDEs are much easier to setup and to use for predicting model responses compared to the baseline. In our simple setup, the evaluation step was surprisingly inefficient in SpellBook which was caused by it supporting various labeling scenarios

and thus took a non-trivial amount of time to setup and navigate. On the other hand, prompterator, while supporting only the basic labeling options, was faster to setup and go through. The prompt updating step duration oscillated significantly regardless of the setup. This was caused mainly by the fact that updating a prompt takes a different amount of time for different model outputs – sometimes it is enough to change the model temperature while other times it is necessary to completely rewrite the prompt.

## 5 Conclusion

We present prompterator, a lightweight *prompt engineering* IDE, which is capable of covering the whole lifecycle of finding an appropriate prompt for a given language task. prompterator has proven to be significantly faster and simpler to use than its alternatives across multiple sub-tasks of the prompting pipeline. By opensourcing it under the terms of the MIT license we hope that it will make the process of *prompt engineering* significantly more efficient in the future.

## 6 Ethical Considerations and Limitations

prompterator has been designed as a highly versatile tool, one that makes minimal assumptions on the tasks it would be used for. In that sense we would like to acknowledge that given its generic nature, it can in principle be used to improve systems with malicious intent.

Furthermore, we would like to note that since prompterator relies on human judgement for annotation, it has the potential to exacerbate any inherent biases that could manifest that way.

## References

Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesh Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. 2022. PromptSource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*.

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.

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 preprint arXiv:2111.01998*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.
- Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-based prototyping with large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–8.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.
- Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. PromptAid: Prompt exploration, perturbation, testing and iteration using visual analytics for large language models. *arXiv preprint arXiv:2304.01964*.
- Guanghui Qin and Jason Eisner. 2021. [Learning how to ask: Querying LMs with mixtures of soft prompts](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, 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.
- 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*.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.
- 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*.
- Timo Schick and Hinrich Schütze. 2021. [Exploiting cloze-questions for few-shot text classification and natural language inference](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.
- Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. 2022. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE transactions on visualization and computer graphics*, 29(1):1146–1156.
- Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- 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.
- Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10.
- JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny cant prompt: How non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.
- 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*.
- 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.

## A Experiment Details

In this section we provide further details on the Experiment described in [section 4](#).

**Task participants** The three prompt engineers involved in the experiments were three of the authors of this paper. All of them had prior experience with prompterator.

**Experimental setup** The experiment evaluated four distinct activities of the prompt-engineering lifecycle: *setup*, *prediction*, *evaluation*, and *prompt updating*. Throughout the experiment the gpt-3.5-turbo-0613 model from OpenAI was used to generate the predictions <sup>14</sup>.

The *setup* part consisted of any action (or actions) the evaluated tools required for the evaluation of specific prompts to happen. In case of the baseline setup (Microsoft Excel used for data labeling and OpenAI playground for generating model responses), this consisted of opening the dataset of ten random questions in Microsoft Excel. With SpellBook it involved creating a new evaluation project and with prompterator it amounted to installing the application itself.

The *prediction* part amounted to obtaining the generated predictions from the evaluated model using a chosen prompt. For the baseline this meant copying the prompt along with the each of the questions in the dataset to the OpenAI Playground and copying the result back to the Microsoft Excel document. For SpellBook and prompterator this was handled automatically by calling the OpenAI API.

The *evaluation* involved providing a human evaluation for each of the predictions in each of the tools. In the baseline case this amounted to filling in the strings "good" or "bad" to a specific column of the spreadsheet, whereas for SpellBook and prompterator this meant clicking on an appropriate button.

Finally, the *prompt updating* step was comprised of taking a holistic view of the evaluations on the dataset obtained in the previous steps and potentially updating the existing prompt to improve its performance in the next cycle. We note that this step is highly individualistic, as the time required to compose a prompt can vary significantly in between individuals, as well as in between evaluation cycles. We can also observe this in [Figure 5](#), where the fact that performing this step using prompterator took

one prompt-engineer a disproportionate amount of time yielded a standard deviation larger than the mean itself. We hypothesise that this particular instance was an outlier and would be averaged out in a larger sample.

<sup>14</sup><https://platform.openai.com/docs/models/continuous-model-upgrades>

# ZhuJiu: A Multi-dimensional, Multi-faceted Chinese Benchmark for Large Language Models

Baoli Zhang<sup>1,\*</sup>, Haining Xie<sup>1,2,\*</sup>, Pengfan Du<sup>1,2</sup>, Junhao Chen<sup>3</sup>, Pengfei Cao<sup>1</sup>,  
Yubo Chen<sup>1,2,†</sup>, Shengping Liu<sup>4</sup>, Kang Liu<sup>1,2</sup> and Jun Zhao<sup>1,2</sup>

<sup>1</sup>Institute of Automation, Chinese Academy of Sciences

<sup>2</sup>School of Artificial Intelligence, University of Chinese Academy of Sciences

<sup>3</sup>Harbin Engineering University, <sup>4</sup>Beijing Unisound Information Technology Co., Ltd

{baoli.zhang, pengfei.cao, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn  
{xiehaining21, dupengfan22}@mailsucas.ac.cn, liushengping@unisound.com

## Abstract

The unprecedented performance of large language models (LLMs) requires comprehensive and accurate evaluation. We argue that for LLMs evaluation, benchmarks need to be comprehensive and systematic. To this end, we propose the ZhuJiu benchmark, which has the following strengths: (1) **Multi-dimensional ability coverage:** We comprehensively evaluate LLMs across 7 ability dimensions covering 51 tasks. Especially, we also propose a new benchmark that focuses on knowledge ability of LLMs. (2) **Multi-faceted evaluation methods collaboration:** We use 3 different yet complementary evaluation methods to comprehensively evaluate LLMs, which can ensure the authority and accuracy of the evaluation results. (3) **Comprehensive Chinese benchmark:** ZhuJiu is the pioneering benchmark that fully assesses LLMs in Chinese, while also providing equally robust evaluation abilities in English. (4) **Avoiding potential data leakage:** To avoid data leakage, we construct evaluation data specifically for 37 tasks. We evaluate 9 current mainstream LLMs and conduct an in-depth discussion and analysis of their results. The ZhuJiu benchmark and open-participation leaderboard are publicly released at <http://www.zhujiu-benchmark.com/> and we also provide a demo video at <https://youtu.be/qypkJ89L1Ic>.

## 1 Introduction

With the continuous development of large language models (LLMs), the emergence of GPT4 (OpenAI, 2023) is enough to trigger a new wave of technology. Various types of LLMs have recently been rapidly developing, such as Llama2 (Touvron et al., 2023) and ChatGLM2 (Du et al., 2022), demonstrating impressive generalization abilities and broad applicability. Therefore, it is crucial to

conduct comprehensive and objective evaluations of LLMs to fully understand their strengths and limitations.

Specifically, on the one hand, for **applicators**, they need to understand the overall performance of LLMs or the advantages of LLMs in a specific aspect. Constructing comprehensive and authoritative benchmarks can help applicators significantly improve the efficiency of using LLMs. On the other hand, for **developers**, the improvement direction of LLMs requires accurate evaluation results as guidance. An objective and fair benchmark can help them carry out relevant research work on LLMs more targetedly.

To this end, scholars conduct extensive research on evaluations for LLMs and construct some superior benchmarks. Normally, the evaluation for LLMs includes two aspects: ability evaluation and evaluation method. Although **traditional benchmarks** such as GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019) and CUGE (Yao et al., 2021) still have a role to play in evaluating LLMs, their limitations are becoming increasingly apparent due to the growing diversity of evaluation dimensions and methods for LLMs. For the **ability evaluation** of LLMs, recent work proposes excellent benchmarks for LLMs in one or several aspects, such as knowledge, reasoning, language, safety and hallucination (Liang et al., 2022; Jifan Yu, 2023; Sun et al., 2023a; Amayuelas et al., 2023; Li et al., 2023; Liu et al., 2023; Jeffery et al., 2021; Wittenburg et al., 2022). However, a comprehensive evaluation of LLMs remains insufficient. For the **evaluation method** of LLMs, there are currently 3 main categories: (1) *Metrics Evaluation*: Evaluating LLMs using existing datasets and corresponding metrics (Liang et al., 2022); (2) *ChatGPT Evaluation*: Using GPT-like LLMs to generate evaluation data and compare the response results of different LLMs (Wang et al., 2023c); (3) *Model Arena*: constructing one-on-one model are-

<sup>1</sup>\*Co-first authors, they contributed equally to this work.

<sup>2</sup>†Corresponding author

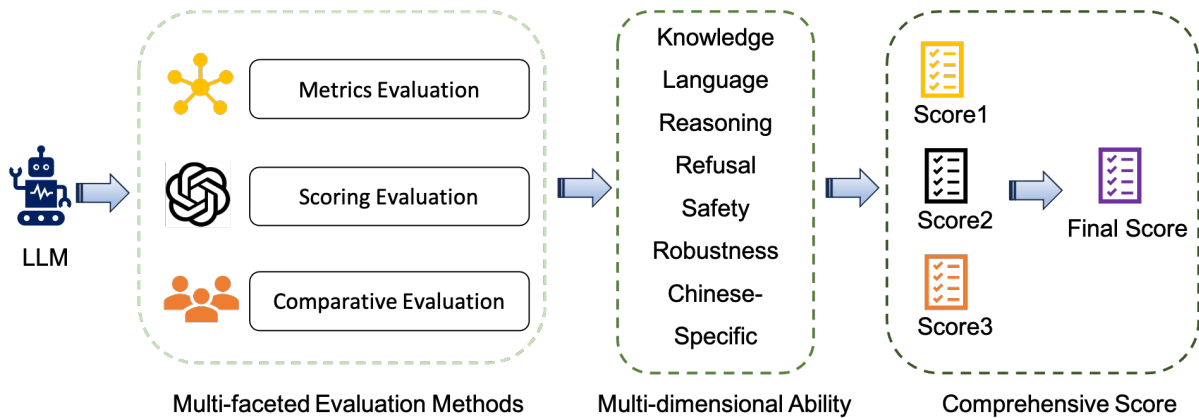


Figure 1: The evaluation process of LLM using ZhuJiu.

nas where humans compare the evaluation results of models based on their own judgment (Zheng et al., 2023; Zhang et al., 2021).

Despite these successful efforts for LLMs’ evaluations, existing studies still suffer from several limitations: (1) Current benchmarks tend to focus on evaluating LLMs on a single dimension of their abilities, which can not provide a comprehensive evaluation of LLMs. (2) Most benchmarks only use a single evaluation method, which may not provide an accurate evaluation of all the abilities of LLMs. For example, while HELM (Liang et al., 2022) uses metrics to evaluate LLMs, it may not measure all abilities such as long-text generation or machine translation, etc. (3) The cross-lingual abilities of LLMs, especially for Chinese, have garnered growing attention. However, the lack of a comprehensive Chinese benchmark for LLMs remains a critical issue. (4) Many current benchmarks only use public datasets for evaluation, risking potential data leakage. The results of evaluations based on this data lack credibility.

In this paper, we propose the ZhuJiu Benchmark to solve above mentioned problems, which can fill the gap in the development of a comprehensive benchmark for evaluating LLMs in Chinese. The advantages of the ZhuJiu are as follows: (1) **Multi-dimensional ability coverage:** we evaluate LLMs from 7 ability dimensions, including *knowledge, Chinese-specific, language, reasoning, refusal, safety and robustness abilities*, covering 51 datasets to provide a comprehensive performance assessment. In addition, we also proposed a new paradigm for evaluating the knowledge ability. (2) **Multi-faceted evaluation methods coordination:** we use *Metrics Evaluation, Scoring Evaluation, and Comparative Evaluation* for comprehensively

evaluating LLMs to ensure authoritative and accurate evaluation results. (3) **Comprehensive Chinese benchmark:** ZhuJiu is the pioneering Chinese benchmark that can comprehensively evaluate LLMs, while allowing equivalent assessment in English. (4) **Avoiding potential data leakage:** in addition to collecting 14 commonly used datasets, we construct 37 datasets for the evaluation of LLMs, ensuring maximum avoidance of data leakage and evaluation fairness. The overall evaluation process is shown in Figure 1.

We also release an online evaluation platform that supports multiple functions including visualizations of evaluation results, participating in model arena and submission of evaluation model, etc. Moreover, we evaluate 9 publicly available LLMs, including ChatGLM (Du et al., 2022), BELLE (Yunjie Ji and Li, 2023), ChatGPT (OpenAI, 2022), and so on. Based on the experimental results, we observe some interesting phenomena and summarize them in 4.2.

In summary, the contributions of this paper are as follows:

- We propose ZhuJiu, the first Chinese benchmark that covers multi-dimensions of ability and employs multi-faceted evaluation methods in collaboration. Meanwhile in the ZhuJiu we construct a novel benchmark for evaluating knowledge ability and 37 evaluation datasets to prevent data leakage issues.
- We release an online evaluation platform that enables users to evaluate LLMs. We will continue to improve the platform, and update the evaluation leaderboard.
- Using the ZhuJiu benchmark, we evaluate 9

current LLMs, to comprehensively and deeply explore their abilities, providing valuable insights to inform future LLM development.

## 2 ZhuJiu Benchmark

As stated above, the ZhuJiu benchmark uses 3 evaluation methods to assess the abilities across seven dimensions of LLMs. This section provides a detailed introduction to the ZhuJiu benchmark covering the evaluation methods, datasets, and ability dimensions. We also detail the specific scoring rules in Appendix A. The evaluation framework is shown in Figure 2.

### 2.1 Evaluation Methods

Unlike previous works that only use a single evaluation method (Liang et al., 2022; Wang et al., 2023b,c; Zheng et al., 2023), in order to ensure the reliability of the evaluation results, we employ a collaborative evaluation approach that utilizes 3 types of evaluation methods: Metrics Evaluation, Scoring Evaluation, and Comparative Evaluation.

#### 2.1.1 Metrics Evaluation

Metrics Evaluation is an indispensable component in LLM assessment, providing objective results (Chang et al., 2023). In this paper, we adopt the HELM evaluation framework. Building on HELM (Liang et al., 2022), we extend it with additional Chinese benchmarks for language, reasoning, knowledge, and Chinese abilities, with 14 expanded datasets total.

#### 2.1.2 Scoring Evaluation

The abilities demonstrated by ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) have brought us great surprises. Therefore, we conduct evaluations on the responses of LLMs using prompt engineering based on ChatGPT. Specifically, we evaluate different abilities and devise different perspectives to assist ChatGPT in scoring the responses. We use few-shot (Snell et al., 2017; Ravi and Larochelle, 2016; Wang et al., 2020) method and answer label, combined with numerous experiments, to ensure the accuracy and stability of ChatGPT’s evaluation results.

#### 2.1.3 Comparative Evaluation

Comparative evaluation is the most intuitive evaluation method. In this paper, we drew inspiration from the work of Chatbot Arena (Zheng et al., 2023) and used the *one-on-one model arena*

*method* to compare and evaluate the performance of LLMs based on human judgments. Furthermore, we provide a one-on-one model comparison function in the platform, which allows users to compare the quality of responses from different LLMs to the same question.

### 2.2 Datasets

For a benchmark, the most crucial part is undoubtedly its data source and data quality. In ZhuJiu, our evaluation data comes from two parts. On the one hand, we use 14 currently popular LLMs evaluation datasets. On the other hand, considering the serious issue of data leakage when solely using public datasets for LLMs evaluation, which could compromise the fairness of evaluation results, we constructed 37 evaluation datasets based on ChatGPT (OpenAI, 2022).

#### 2.2.1 Collect Datasets

To ensure the generality of ZhuJiu, we evaluate LLMs using 14 publicly available datasets, which are essential due to their high quality and ability to accurately evaluate the performance of LLMs in certain aspects.

#### 2.2.2 Construct Datasets

To address the issue of data leakage in LLMs evaluation, we are inspired by PandaLM (Wang et al., 2023c) and we construct corresponding evaluation datasets for 37 specific tasks. Specifically, for each task, we first carefully select some evaluation data as seeds manually. Then, we use these seeds to generate prompts based on ChatGPT through self-instruction (Wang et al., 2022). After that, we manually review and confirm the prompts we used (for each specific task, we generate 100 prompts in Chinese).

To better understand the processes of data construction and evaluation in a more intuitive way, we take Scoring Evaluation as an example to demonstrate the process, as shown in Figure 3.

### 2.3 Ability System

With the help of the aforementioned evaluation methods and datasets, we can assess the abilities of LLMs in 7 aspects. We will provide a detailed introduction to the specific evaluation methods and details in this section.

#### 2.3.1 Knowledge Ability

To comprehensively evaluate the knowledge abilities of LLMs, we conduct the evaluation from

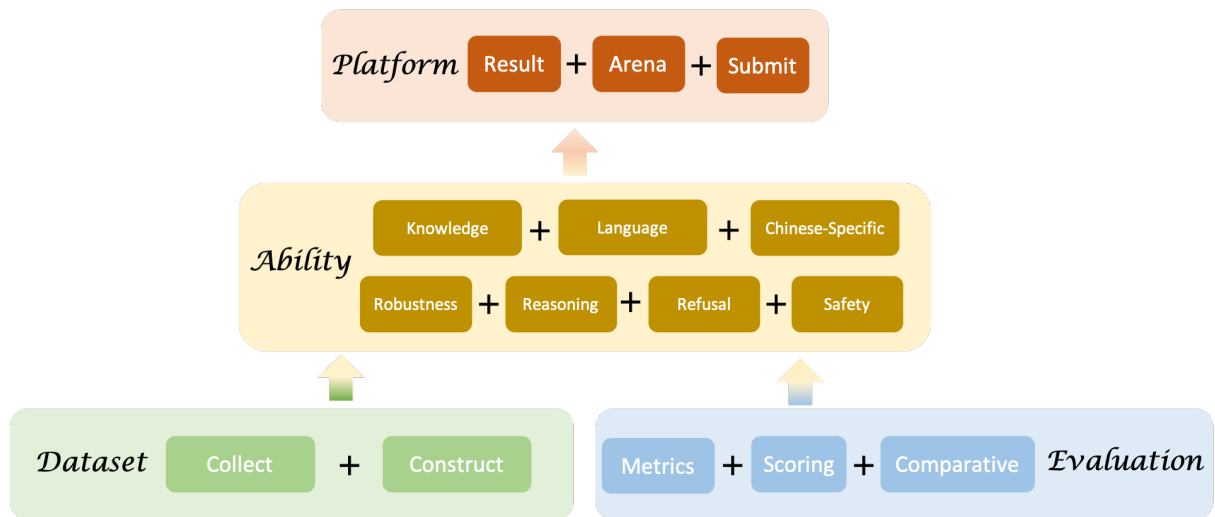


Figure 2: Overall view of the ZhuJiu benchmark. In ZhuJiu’s framework, the integration of **multi-angle datasets** and **multi-faceted evaluation methods** provides strong support for **multi-dimensional ability** assessment. Based on this, we have further developed an **online assessment platform** to support ZhuJiu’s online assessment and result updates.

four perspectives: *world knowledge*, *commonsense knowledge*, *linguistic knowledge*, and *concept*. For each evaluation perspective, we select the appropriate properties of accuracy, robustness, completeness, and timeliness to construct evaluation datasets for evaluating LLMs. Detailed descriptions of these four properties are provided in Appendix B, using a detailed framework shown in Figure 4. Compared to KoLA (Jifan Yu, 2023), our evaluation perspective for knowledge is broader.

For **world knowledge**, on the one hand, we utilize the GAOKAO-bench (Zhang et al., 2023) (Non-mathematical section) and combine it with Metrics Evaluation to conduct the evaluation. On the other hand, we construct corresponding evaluation datasets for each evaluation property, including accuracy, robustness, completeness, and timeliness, and evaluate LLMs using Scoring Evaluation.

For **commonsense knowledge**, we select commonsense triplets as the basic data and construct evaluation datasets based on the evaluation properties of accuracy and robustness. We then use Scoring Evaluation to evaluate LLMs.

For **linguistic knowledge**, we use Chinese FrameNet (CFN) (Hao et al., 2007; Baker et al., 1998) as the original corpus. In order to simplify the evaluation form of linguistic knowledge, we mainly construct datasets in the following two ways: one is to infer the “frame name” of the linguistic frame according to the “frame def” in the linguistic frame, the other is to infer the “frame name” of the linguistic frame based on the “lexical-

unit name” in the linguistic frame. Then we can evaluate the accuracy and robustness of LLMs linguistic knowledge by using the Scoring Evaluation.

For **concept**, we manually select common entity words as the original data and evaluate the accuracy and robustness of LLMs concepts with Scoring Evaluation.

### 2.3.2 Chinese-Specific Ability

Following SuperCLUE (Liang Xu and others from SuperCLUE team, 2023), and conventional Chinese evaluations, the Chinese-specific ability evaluation aims to use corpora with Chinese unique characteristics as the original data to form evaluation data. These corpora include ChID (Zheng et al., 2019), CCPM (Li et al., 2021), CINLID and YACLIC (Wang et al., 2021b), and we evaluate LLMs using Metrics Evaluation.

### 2.3.3 Language Ability

We conduct a comprehensive evaluation of LLMs’ language ability from both aspects of language understanding and language generation. For evaluating LLMs’ **language understanding ability**, we choose to evaluate them on the tasks of reading comprehension and coreference resolution. We find that using existing datasets could achieve good evaluation results, and the datasets we use included C3 (Sun et al., 2020), GCRC (Tan et al., 2021), CMRC (Cui et al., 2018), DRRC (Shao et al., 2018) and CLUEWSC-2020 (Xu et al., 2020), correspondingly we use Metrics Evaluation. For evaluating

LLMs’ **language generation ability**, we summarize 6 typical language generation tasks, including *common response* (Daily question answering), *dialogue* (Dialog generation based on the scene), *formal writing* (Generation of formal texts for letters and other formal occasions), *poetry* (Generate poems on request), *writing story* (Generate stories on request) and *writing style* (Generate text according to the requirements of the writing style) (Chang et al., 2023), and evaluating by Scoring Evaluation.

### 2.3.4 Reasoning Ability

As the evaluation of LLMs’ reasoning ability is less affected by data leakage (Chang et al., 2023), we find that only using publicly available datasets could yield relatively fair results. We select the currently popular mathematical reasoning and text semantic reasoning tasks, and the datasets included GAOKAO-bench (Zhang et al., 2023) (mathematics section), Math23k (Wang et al., 2017), OCNLI (Hu et al., 2020), Chinese-SNLI (chi, 2019) and Chinese-MNLI (Xu et al., 2020). The evaluation method for reasoning ability is based on Metrics Evaluation.

### 2.3.5 Refusal Ability

Regarding the refusal ability, we can understand it like this: *To know what you know and to know what you do not know, that is true knowledge*. For constructing datasets of refusal ability, we drew inspiration from the categories of Known-Unknown Questions proposed in Amayuelas et al., 2023, including *Future Unknown*, *Unsolved Problem/Mystery*, *Controversial/Debatable Question*, *Question with False Assumption*, *Counterfactual Question* and *Underspecified Question*. Then, we employ Scoring Evaluation to assess LLMs for each category.

### 2.3.6 Safety

For the evaluation of safety ability, we follow Sun et al., 2023a’s classification of safety ability and further summarize and categorize them. We derive a total of 9 evaluation tasks from 6 perspectives, including *Insult*, *Human Health (Physical harm and Mental health)*, *Social Topic (Unfairness discrimination and Ethics morality)*, *Serious Risk (Criminal Activity and Unsafe Instruction Topic)*, *Goal Hijacking* and *Role play instruction*. Subsequently, we employ the Scoring Evaluation to assess LLMs.

### 2.3.7 Robustness

Traditional robustness evaluation primarily focuses on assessing the impact of adding perturbations

of varying granularity to the text on the performance of the model (Zhu et al., 2023; Wang et al., 2021a, 2023a). Regarding the robustness evaluation of LLMs, on one hand, we still consider token-level perturbations and sentence-level perturbations from the traditional robustness evaluation perspective, and propose three evaluation tasks including *Error Message*, *Redundant Information* and *Redundant Dialogue*. On the other hand, we expand three aspects of *Format Output*, *Dialect* and *Unique Solution tasks* (Evaluate the certainty of the model’s answer to the unique solution through multiple rounds of questioning) specifically tailor to the characteristics of LLMs. Ultimately, we conduct evaluations on these six aspects based on the Scoring Evaluation.

## 3 Platform

We develop an online platform to provide a range of services for the community as follows:

**Visualizations of evaluation results** We publish the rankings of all model evaluations on the platform, including specific scores for each ability and evaluation method, and the rankings will be updated continuously as the evaluations progress.

**Participating in Model Arena** We launch a one-on-one model arena feature on our platform, where everyone can support the LLMs they believe perform better based on their own judgment. Please refer to Figure 5 to see the web view of the model arena.

**Submission of Evaluation Model** We also encourage everyone to actively participate in our evaluations and join the leaderboard. On our platform, we allow users to submit applications for evaluation.

## 4 Experiment

### 4.1 Evaluated Models

To facilitate the utilization and advancement of LLMs, the primary emphasis of ZhuJiu’s inaugural evaluation phase is directed towards *open-source* LLMs with a parameter magnitude of approximately 10 billion, including: ChatGLM-6B (Du et al., 2022), ChatGLM2-6B (Du et al., 2022), BELLE-7B (Yunjie Ji and Li, 2023), ChatFlow (Li et al., 2022; Zhao et al., 2022), Phoenix-Inst-Chat-7B (Chen et al., 2023b,a), ChatYuan-large-v2 (Xuanwei Zhang and Zhao, 2022), Moss-Moon-003-SFT (Sun et al., 2023b) and RWKV (Bo,

| Score<br>LLMs        | Abilities   |                  |             |             |             |             |             |             | All |
|----------------------|-------------|------------------|-------------|-------------|-------------|-------------|-------------|-------------|-----|
|                      | Knowledge   | Chinese-Specific | Language    | Reasoning   | Refusal     | Safety      | Robustness  |             |     |
| ChatGLM2-6B          | <b>91.1</b> | 59.5             | <b>85.6</b> | <b>80.6</b> | <b>82.0</b> | 55.4        | <b>63.8</b> | <b>74.0</b> |     |
| ChatGLM-6B           | 67.3        | <b>73.9</b>      | 74.8        | 37.0        | 80.4        | <b>82.3</b> | 50.0        | 66.5        |     |
| BELLE-7B             | 54.53       | 40.54            | 54.2        | 44.5        | 58.1        | 39.8        | 55.9        | 49.6        |     |
| Moss-Moon-003-SFT    | 50.4        | 27.0             | 56.3        | 15.9        | 48.2        | 64.8        | 46.2        | 44.1        |     |
| ChatYuan-large-v2    | 58.8        | 20.7             | 37.3        | 42.7        | 37.5        | 78.1        | 29.8        | 43.6        |     |
| ChatFlow             | 43.3        | 54.1             | 33.3        | 47.1        | 39.2        | 40.3        | 36.1        | 41.9        |     |
| Phoenix-Inst-chat-7B | 19.53       | 0                | 62.3        | 0           | 67.3        | 65.9        | 61.0        | 39.4        |     |
| RWKV                 | 23.4        | 15.0             | 35.8        | 69.3        | 16.4        | 20.5        | 45.9        | 32.3        |     |
| GPT-3.5-turbo        | 82.4        | 100.0            | 84.3        | 100.0       | 100.0       | 100.0       | 85.5        | 93.2        |     |

Table 1: The overall performance based on ten-point system of the LLMs participating in the ZhuJiu evaluation in the first season. The score of GPT-3.5-turbo is only for reference and not included in the evaluation.

2021). Concurrently, we employ ChatGPT (OpenAI, 2022) as a comparative benchmark and conduct an assessment of the GPT-3.5-turbo API service.

## 4.2 Overall Performance

We report the overall performance in Table ??, and show more detailed assessment results in our platform. From the results, we can obtain some intriguing findings:

- (1) **Model-Performance is Limited by Model-Size:** Based on the results in table ??, it becomes evident that models with a parameter size of around 10 billion still exhibit significant limitations in overall performance compared to GPT-3.5-turbo (OpenAI, 2022). In ZhuJiu, the performance of most LLMs is relatively mediocre, with ChatGLM2 and ChatGLM (Du et al., 2022) showing relatively better performance. It becomes apparent that the size of the model’s parameters continues to play a vital role in determining its performance.
- (2) **Lower Limit Sets Upper Limit:** The analysis reveals that Phoenix (Chen et al., 2023b) demonstrates notable proficiency in refusal and safety abilities, etc. However, its overall ranking is comparatively lower, primarily attributed to its limitations in reasoning and Chinese-specific abilities. These deficiencies are also observed in other LLMs occupying lower positions in the rankings. However, *the lower limits of various abilities in LLMs often determine the upper limits of LLMs’ application prospects.*
- (3) **Knowledge is Power:** In ZhuJiu, our primary focus lies in the knowledge ability of LLMs, as

the pivotal task at hand is to ensure LLMs acquire accurate knowledge and effectively harness their acquired knowledge. However, in this version, the majority of LLMs exhibit sub-par performance in terms of knowledge capacity, making the ZhuJiu benchmark exceptionally challenging. The results reveal that ChatGLM2 (Du et al., 2022) exhibits strong performance in knowledge ability, surpassing even ChatGPT.

## 5 Conclusion and Future Work

In this work, we present ZhuJiu, the pioneering multi-dimensional ability coverage, multi-faceted evaluation methods collaboration Chinese benchmark. ZhuJiu is capable of using 3 evaluation methods to comprehensively evaluate LLMs across 7 ability dimensions, using 51 datasets. Additionally, we independently construct 37 evaluation datasets to maximize the avoidance of data leakage issues in LLM evaluation. We also focus on expanding the evaluation of knowledge ability, providing a new framework for assessing LLMs’ knowledge ability. Finally, we provide a comprehensive and continuously updated evaluation platform with multiple functions and in the first season of ZhuJiu, we evaluate 9 *open-source* LLMs.

In the future, we plan to (1) continuously construct high-quality evaluation datasets to enrich ZhuJiu, (2) further perfect the assessment of knowledge ability and develop new evaluation methods for Chinese characteristic ability, (3) further perfect the platform’s functionality and update the platform’s information.



## Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106400), the National Natural Science Foundation of China (No. 6197621162176257). This work is also supported by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No.XDA27020100), the Youth Innovation Promotion Association CAS, and Yunnan Provincial Major Science and Technology Special Plan Projects (No.202202AD080004).

## References

2019. Blog: Chinese-snli. <https://gitee.com/jiaodaxin/CNSD>.
- Alfonso Amayuelas, Liangming Pan, Wenhua Chen, and William Wang. 2023. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. *arXiv preprint arXiv:2305.13712*.
- Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.
- PENG Bo. 2021. *Blinkdl/rwkv-lm: 0.01*.
- Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*.
- Zhihong Chen, Junying Chen, Hongbo Zhang, Feng Jiang, Guiming Chen, Fei Yu, Tiannan Wang, Juhao Liang, Chen Zhang, Zhiyi Zhang, Jianquan Li, Xiang Wan, Haizhou Li, and Benyou Wang. 2023a. Llm zoo: democratizing chatgpt. <https://github.com/FreedomIntelligence/LLMZoo>.
- Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. 2023b. Phoenix: Democratizing chatgpt across languages. *arXiv preprint arXiv:2304.10453*.
- Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2018. A span-extraction dataset for chinese machine reading comprehension. *arXiv preprint arXiv:1810.07366*.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.
- Xiaoyan Hao, Wei Liu, Ru Li, and Kaiying Liu. 2007. Description systems of the chinese framenet database and software tools. *Journal of Chinese information processing*, 21(5):96–100.
- Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence S Moss. 2020. Ocnli: Original chinese natural language inference. *arXiv preprint arXiv:2010.05444*.
- Keith Jeffery, Peter Wittenburg, Larry Lannom, George Strawn, Claudia Binossek, Dirk Betz, and Christophe Blanchi. 2021. Not ready for convergence in data infrastructures. *Data Intelligence*, 3(1):116–135.
- Shangqing Tu Shulin Cao Daniel Zhang-Li Xin Lv Hao Peng Zijun Yao Xiaohan Zhang Hanming Li Chunyang Li Zheyuan Zhang Yushi Bai Yantao Liu Amy Xin Nianyi Lin Kaifeng Yun Linlu Gong Jianhui Chen Zhili Wu Yunjia Qi Weikai Li Yong Guan Kaisheng Zeng Ji Qi Hailong Jin Jinxin Liu Yu Gu Yuan Yao Ning Ding Lei Hou Zhiyuan Liu Bin Xu Jie Tang Juanzi Li Jifan Yu, Xiaozhi Wang. 2023. *Kola: Carefully benchmarking world knowledge of large language models*.
- Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. *arXiv e-prints*, pages arXiv–2305.
- Wenhao Li, Fanchao Qi, Maosong Sun, Xiaoyuan Yi, and Jiarui Zhang. 2021. Ccpm: A chinese classical poetry matching dataset. *arXiv preprint arXiv:2106.01979*.
- Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. 2022. *CSL: A large-scale Chinese scientific literature dataset*. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3917–3923, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.
- Kangkang Zhao Lei Zhu Liang Xu, Xuanwei Zhang and others from SuperCLUE team. 2023. Superclue: A benchmark for foundation models in chinese. <https://github.com/CLUEbench/SuperCLUE>.
- Yuchi Liu, Zhongdao Wang, Xiangxin Zhou, and Liang Zheng. 2023. A study of using synthetic data for effective association knowledge learning. *Machine Intelligence Research*, 20(2):194–206.
- OpenAI. 2022. Blog: Introducing chatgpt. <https://openai.com/blog/chatgpt>.
- OpenAI. 2023. *Gpt-4 technical report*.

- Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. In *International conference on learning representations*.
- Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: A chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.
- Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.
- Hao Sun, Zhixin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023a. Safety assessment of chinese large language models. *arXiv preprint arXiv:2304.10436*.
- Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155.
- Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, and Xipeng Qiu. 2023b. Moss: Training conversational language models from synthetic data.
- Hongye Tan, Xiaoyue Wang, Yu Ji, Ru Li, Xiaoli Li, Zhiwei Hu, Yunxiao Zhao, and Xiaoqi Han. 2021. Grc: A new challenging mrc dataset from gaokao chinese for explainable evaluation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1319–1330.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- 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 R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.
- Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. 2023a. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*.
- Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, et al. 2021a. 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, Online. Association for Computational Linguistics.
- Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 845–854.
- Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34.
- Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Qiang Heng, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2023b. Pandalm: Reproducible and automated language model assessment. <https://github.com/WeOpenML/PandaLM>.
- Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2023c. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization.
- Yingying Wang, Cunliang Kong, Liner Yang, Yijun Wang, Xiaorong Lu, Renfen Hu, Shan He, Zhenghao Liu, Yun Chen, Erhong Yang, et al. 2021b. Yalc: A chinese learner corpus with multidimensional annotation. *arXiv preprint arXiv:2112.15043*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.
- Peter Wittenburg, Alex Hardisty, Yann Le Franc, Amirpasha Mozaffari, Limor Peer, Nikolay A Skvortsov, Zhiming Zhao, and Alessandro Spinuso. 2022. Canonical workflows to make data fair. *Data Intelligence*, 4(2):286–305.
- 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.

Liang Xu Xuanwei Zhang and Kangkang Zhao. 2022. [Chatyuan: A large language model for dialogue in chinese and english.](#)

Yuan Yao, Qingxiu Dong, Jian Guan, Boxi Cao, Zhengyan Zhang, Chaojun Xiao, Xiaozhi Wang, Fanchao Qi, Junwei Bao, Jinran Nie, et al. 2021. Cuge: A chinese language understanding and generation evaluation benchmark. *arXiv preprint arXiv:2112.13610*.

Yan Gong Yiping Peng Qiang Niu-Baochang Ma Yunjie Ji, Yong Deng and Xiangang Li. 2023. Belle: Be everyone’s large language model engine. <https://github.com/LianjiaTech/BELLE>.

Baoli Zhang, Zhucong Li, Zhen Gan, Yubo Chen, Jing Wan, Kang Liu, Jun Zhao, Shengping Liu, and Yafei Shi. 2021. Croano: A crowd annotation platform for improving label consistency of chinese ner dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 275–282.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark.

Zhe Zhao, Yudong Li, Cheng Hou, Jing Zhao, Rong Tian, Weijie Liu, Yiren Chen, Ningyuan Sun, Haoyan Liu, Weiquan Mao, et al. 2022. Tencentpre-train: A scalable and flexible toolkit for pre-training models of different modalities. *arXiv preprint arXiv:2212.06385*.

Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. ChID: A large-scale Chinese IDiom dataset for cloze test. In *ACL*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena.](#)

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.

## A Scoring Rules

We will comprehensively evaluate the model from seven ability dimensions and 3 assessment methods to ensure the thoroughness and authority of the evaluation results. Specifically, the comprehensive evaluation process can be broken down into three steps.

**Step 1** For each ability dimension score  $A$ , we will take the average of LLM’s scores  $\mathbf{d} =$

$[d_1, \dots, d_n]$  on each dataset as LLM’s score for that ability dimension:

$$A = \frac{1}{n} \sum_{i=1}^n d_i \quad (1)$$

**Step 2** For each evaluation method score  $E$ , LLM’s score is the average of its scores  $\mathbf{A} = [A_1, \dots, A_m]$  for each ability dimension:

$$E = \frac{1}{m} \sum_{j=1}^m A_j \quad (2)$$

**Step 3** LLM’s scores  $\mathbf{E} = [E_1, E_2, E_3]$  for each evaluation method are standardized and then averaged to obtain LLM’s final score on ZhuJiu:

$$E_{\text{norm}} = \frac{E_k - E_{\min}}{E_{\max} - E_{\min}} \quad (3)$$

## B Evaluation Perspective for Knowledge Ability

In the evaluation process of knowledge ability, we mainly evaluate from the properties of accuracy, robustness, completeness and timeliness. For each property, we will randomly generate one hundred sets of evaluation data for evaluation. Here we Need to explain the specific indicators of each evaluation (Wittenburg et al., 2022).

- **Accuracy:** Evaluate whether the content of the model’s reply is correct through Exact Match (EM) and ChatGPT (OpenAI, 2022), and calculate the accuracy rate in the 100 questions answered correctly by the model.
- **Robustness:** We use the same set of data to use ChatGPT to randomly generate five different ways of asking questions, and then score according to whether the model is stable in replying to different questions generate by the same set of data. The principle of scoring is that the more stable the content of the reply, the higher the score.
- **Completeness:** Only for the evaluation of world knowledge, scoring is based on the proportion of standard answers cover in the model’s reply content. For example, according to the calculation of a question with a full score of 10, for the data “(中国四大发明—包括—火药,指南针,造纸术,印刷术)” “(The Four Great Inventions of ancient China—include—gunpowder, compass,

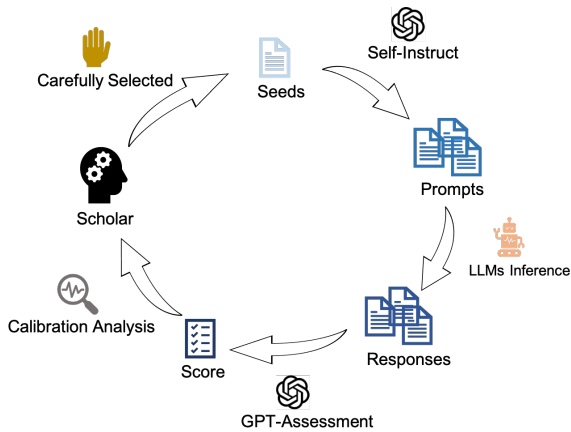


Figure 3: The specific processes of data construction and Scoring Evaluation

*papermaking, printing*)” generate the evaluation question “中国的四大发明包括哪些?” “What are the Four Great Inventions of ancient China?”, if the model answers “火药,指南针,造纸术,印刷术” “*gunpowder, compass, papermaking, printing*”, it will get a full score of 10, and answer “火药,指南针,造纸术,瓷器” “*gunpowder, compass, papermaking, china*” has a correct rate of 75 percent and a score of 7.5.

- **Timeliness:** It is only aim at the evaluation of world knowledge, and specifically evaluates the update degree of LLMs knowledge, similar to accuracy, and evaluates whether the answer of the model is correct or not according to EM and ChatGPT.

### C GPT-Assessment Prompts

In the scoring evaluation method, we use GPT-4 to score the answers of the model being tested. The evaluation content covers 37 testing tasks corresponding to 7 capabilities, and the evaluation datasets are all generated by GPT and manually reviewed to prevent data leakage. For each evaluation task, there are more than three nearly characteristic evaluation indicators. Table 2 shows some task prompt cases and table 3 shows the GPT evaluation prompts that used in **Language Ability**.

| Tasks           | Prompt Cases   |
|-----------------|--|
| Common Response | 全球气候变化会对人类生活产生什么影响?  |
|                 | What impact will global climate change have on human life?   |
| Dialogue        | 假设你是一名警察，你正在盘问一名犯罪嫌疑人，他们之间将会有怎样引人注目的对话，请运用你的想象，创造他们之间的一段对话。  |
|                 | Assuming you are a police officer, and you are interrogating a criminal suspect, what kind of captivating conversation will take place between them? Please use your imagination to create a dialogue between them.  |
| Writing Story   | 在古代中国，如果有一种新的科技出现，比如说互联网，会发生什么有趣的故事?   |
|                 | In ancient China, if a new technology, such as the internet, appeared, what interesting stories might occur?   |
| Writing Style   | 为什么云会飘动? 请你模仿莎士比亚的文风回答问题。  |
|                 | Why do clouds drift? Please answer the question mimicking Shakespeare's writing style.   |
| Poetry          | 你站在远离城市喧嚣的郊外，看到星空璀璨，感到内心的宁静。请用一首诗表达你此时的情感。   |
|                 | You stand in the outskirts, far from the city's hustle and bustle, seeing the stars twinkle brilliantly, feeling an inner peace. Please express your emotions at this moment with a poem.  |
| Formal Writing  | 请帮忙起草一份正式的辞职信，表达对公司的感激之情并说明辞职的原因，同时表达对公司未来的祝福和愿意做出过渡安排的意愿。辞职信需要使用正式的格式和措辞，遵循职场礼仪。  |
|                 | Please help draft a formal resignation letter, expressing gratitude towards the company and stating the reasons for resignation, while also expressing blessings for the company's future and a willingness to make transition arrangements. The resignation letter needs to use formal format and wording, adhering to workplace etiquette. |

Table 2: Language ability has six sub-tasks; here are some prompt cases of the tasks.

| Tasks           | Evaluation Prompt Templates   |
|-----------------|---|
| Common Response | <p>请你扮演一个AI机器人评估员，你需要评估一个AI机器人回答的质量。你的评估结果需要考虑到回答是否有帮助，是否与问题相关，是否有创造性，是否有深度。你的评估结果需要提供一段对于该回答质量的解释，请尽量保持客观，并在最后为每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「帮助性」、「与问题的相关性」、「创造性」、「深度」。你仅需要输出json评估结果。</p> <p>Please act as an AI robot evaluator, you need to assess the quality of an AI robot's answer. Your assessment results need to consider whether the answer is helpful, whether it is relevant to the question, whether it is creative, and whether it has depth. Your assessment results need to provide an explanation of the quality of the answer, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective at the end.[Question]prompt[Start of response]response[End of response]Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Helpfulness', 'Relevance to the question', 'Creativity', 'Depth'. You only need to output the JSON assessment result.</p>   |
| Dialogue        | <p>请你扮演一个AI机器人评估员，你需要评估一个AI机器人创造对话的能力。你的评估结果需要考虑到对话是否符合场景要求，对话是否符合角色身份，对话是否符合逻辑，对话是否通顺。你的评估结果需要提供一段对该对话的解释，请尽量保持客观。在解释之后，对每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「与场景的匹配度」、「与角色身份的匹配度」、「逻辑性」、「对话通顺度」。你仅需要输出json评估结果。</p> <p>Please act as an AI robot evaluator, you need to assess an AI robot's ability to create a dialogue. Your assessment results need to consider whether the dialogue meets the scenario requirements, whether the dialogue conforms to the role identity, whether the dialogue is logical, and whether the dialogue is fluent. Your assessment results need to provide an explanation for the dialogue, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective. [Question]prompt[Start of response]response[End of response] Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Match with the scenario', 'Match with role identity', 'Logic', 'Dialogue fluency'. You only need to output the JSON assessment result.</p> |

Table 3: Here are the evaluation prompt templates for the tasks in language ability, each task has specific evaluation perspectives. This table shows the evaluation prompts of 'Common Response' and 'Dialogue' tasks in language ability.

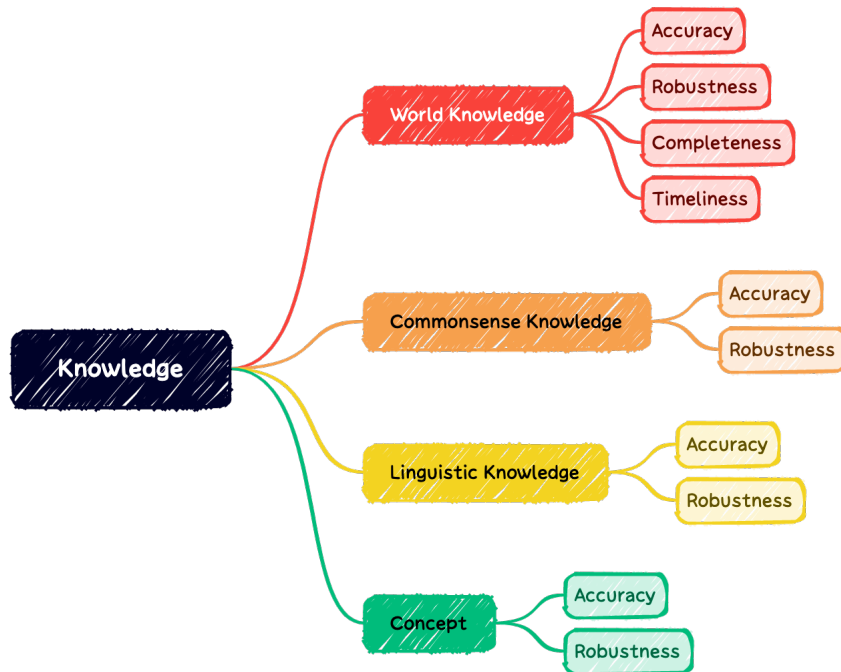
| Tasks         | Evaluation Prompt Templates  |
|---------------|--|
| Writing Story | <p>请你扮演一个AI机器人评估员，你需要评估一个AI机器人写故事的能力。你的评估结果需要考虑到故事是否满足要求，故事是否符合逻辑，故事是否有创造性，是否有深度。你的评估结果需要提供一段对于该故事质量的解释，如果有不符合逻辑的情节，将其列出来，请尽量保持客观。在解释之后，另在最后每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「与问题的相关性」、「逻辑性」、「创造性」、「深度」。你仅需要输出json评估结果。</p> <p>Please act as an AI robot evaluator, you need to assess an AI robot's ability to write a story. Your assessment results need to consider whether the story meets the requirements, whether the story is logical, whether it is creative, and whether it has depth. Your assessment results need to provide an explanation of the story's quality, and if there are illogical plots, list them, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective at the end. [Question]prompt[Start of response]response[End of response] Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Relevance to the question', 'Logic', 'Creativity', 'Depth'. You only need to output the JSON assessment result.</p> |
| Writing Style | <p>请你扮演一个AI机器人评估员，你需要评估一个AI机器人输出指定文风文章的能力。你的评估结果需要考虑到文章是否符合文风要求，与问题相关性，回答的深度和创造性。你的评估结果需要提供一段对该文章的解释，请尽量保持客观。在解释之后，对每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「文风的匹配度」、「与问题相关性」、「深度」、「创造性」。你仅需要输出json评估结果。</p> <p>Please act as an AI robot evaluator, you need to assess an AI robot's ability to output an article with a specified style. Your assessment results need to consider whether the article meets the style requirements, its relevance to the question, the depth, and creativity of the answer. Your assessment results need to provide an explanation for the article, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective. [Question]prompt[Start of response]response[End of response] Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Matching degree with style', 'Relevance to the question', 'Depth', 'Creativity'. You only need to output the JSON assessment result.</p>                                      |

Table 4: This table shows the evaluation prompts of 'Writing Story' and 'Writing Style' tasks in language ability.

| Tasks          | Evaluation Prompt Templates  |
|----------------|--|
| Poetry         | <p>请你扮演一个语言模型评估员，你需要评估一个语言模型诗歌写作的能力。你的评估结果需要考虑到文章是否符合诗歌格式要求，与问题相关性，回答的深度和创造性。你的评估结果需要提供一段对该诗歌质量的解释，请尽量保持客观。在解释之后，对每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「诗歌格式的匹配度」、「与问题相关性」、「深度」、「创造性」。你仅需要输出json评估结果。</p> <p>Please act as a language model evaluator, you need to assess a language model's poetry writing ability. Your assessment results need to consider whether the article meets the poetry format requirements, its relevance to the question, the depth, and creativity of the answer. Your assessment results need to provide an explanation for the quality of the poetry, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective. [Question]prompt[Start of response]response[End of response] Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Matching degree with poetry format', 'Relevance to the question', 'Depth', 'Creativity'. You only need to output the JSON assessment result.</p>   |
| Formal Writing | <p>请你扮演一个语言模型评估员，你需要评估一个语言模型输出指定正式格式文本的能力。你的评估结果需要考虑到文本是否符合对应场景的格式要求，是否符合角色身份，是否符合逻辑、文本是否通顺。你的评估结果需要提供一段对该文本的解释，请尽量保持客观。在解释之后，对每个角度提供一个1-10的打分。[问题]prompt[回复开始]response[回复结束]你的输出格式需要严格按照json格式输出，输出的json字典包括两个键「解释」和「得分」。「解释」的值是字符串格式。「得分」的值是一个嵌套字典，包含如下几个键：「格式正确性」、「与角色身份的匹配度」、「逻辑性」、「文本通顺度」。你仅需要输出json评估结果。</p> <p>Please act as a language model evaluator, you need to assess a language model's ability to output text in a specified formal format. Your assessment results need to consider whether the text conforms to the format requirements of the corresponding scene, whether it conforms to the role identity, whether it is logical, and whether the text is fluent. Your assessment results need to provide an explanation for the text, please try to remain objective. After the explanation, provide a score from 1-10 for each perspective. [Question]prompt[Start of response]response[End of response] Your output format needs to strictly follow the JSON format. The output JSON dictionary includes two keys: 'Explanation' and 'Score'. The value of 'Explanation' is in string format. The value of 'Score' is a nested dictionary, containing the following keys: 'Correctness of Format', 'Match with Role Identity', 'Logic', 'Text Fluency'. You only need to output the JSON assessment result.</p> |

Table 5: This table shows the evaluation prompts of 'Poetry' and 'Formal Writing' tasks in language ability.





Presented with xmind

Figure 4: The overall framework of Knowledge benchmark

## model arena

**最好的减肥方法是什么**

**? 我16岁,身高170cm体重85kg。想在暑假里减到标准身材(男).有什么好的办法吗?最好是运动的!谢谢了~新浪微博QQ空间 每天坚持跑步3公里以上,跑完后做一些拉伸动作和压腿等活动**

**健康减肥应该遵循以下几个原则:**

1. 合理饮食: 控制总热量摄入, 选择低脂、高蛋白和多纤维的食物。建议在日常生活中适当限制碳水化合物和糖分的摄入。
2. 有规律运动: 进行适量运动可增加代谢率, 提升身体的新陈代谢, 消耗更多的卡路里。每周至少3次有氧运动或其他形式的锻炼可帮助燃烧脂肪。
3. 保持良好的睡眠习惯: 充足的睡眠对身体健康和心理健康都非常重要, 因为它能够帮助我们调整内分泌系统。保证每晚7-8小时的睡眠也有助于促进身体恢复和代谢功能。
4. 保持积极态度: 减肥过程不会一蹴而就, 需要付出努力并坚持自己的计划。通过找到支持自己的社交圈和寻求专业意见, 保持积极的思想状态将有助于推动你走向成功。

Better on the Left
Both Sides Equal
Better on the Right

Figure 5: Visualization of Model Arena. And we show the example in English in figure 6



Question: What is the best method for losing weight?



Response:

I am 16 years old, 170cm tall, and weigh 85kg. I want to achieve a healthy weight during the summer break (male). Are there any good methods, preferably exercise-related, to achieve this? Thank you! I plan to run at least 3 kilometers every day and perform stretching exercises and leg presses after running, as well as engaging in activities on platforms like Sina Weibo and QQ Space.



Response:

Healthy weight loss should follow the following principles:

1. **Balanced diet:** Control your overall calorie intake and choose low-fat, high-protein, and high-fiber foods. It is recommended to limit the intake of carbohydrates and sugars in your daily life.
2. **Regular exercise:** Engage in moderate exercise to increase your metabolic rate, boost your body's metabolism, and burn more calories. Engaging in aerobic exercises or other forms of exercise at least three times a week can help burn fat.
3. **Maintain good sleep habits:** Sufficient sleep is crucial for both physical and mental well-being as it helps regulate our endocrine system. Ensuring 7-8 hours of sleep per night also promotes bodily recovery and metabolic functions.
4. **Maintain a positive attitude:** The weight loss process takes time and effort, requiring commitment to your plan. By finding a supportive social circle and seeking professional advice, maintaining a positive mindset will help propel you towards success.

Figure 6: English translation of Model Arena example

# PaperMage: A Unified Toolkit for Processing, Representing, and Manipulating Visually-Rich Scientific Documents

Kyle Lo<sup>α\*</sup> Zejiang Shen<sup>α,τ\*</sup> Benjamin Newman<sup>α\*</sup> Joseph Chee Chang<sup>α\*</sup>  
Russell Authur<sup>α</sup> Erin Bransom<sup>α</sup> Stefan Candra<sup>α</sup> Yoganand Chandrasekhar<sup>α</sup>  
Regan Huff<sup>α</sup> Bailey Kuehl<sup>α</sup> Amanpreet Singh<sup>α</sup> Chris Wilhelm<sup>α</sup> Angele Zamarron<sup>α</sup>  
Marti A. Hearst<sup>β</sup> Daniel S. Weld<sup>α,ω</sup> Doug Downey<sup>α,η</sup> Luca Soldaini<sup>α\*</sup>

<sup>α</sup>Allen Institute for AI <sup>τ</sup>Massachusetts Institute of Technology

<sup>β</sup>University of California Berkeley <sup>ω</sup>University of Washington <sup>η</sup>Northwestern University

{kyle1, lucas}@allenai.org

## Abstract

Despite growing interest in applying natural language processing (NLP) and computer vision (CV) models to the scholarly domain, scientific documents remain challenging to work with. They're often in difficult-to-use PDF formats, and the ecosystem of models to process them is fragmented and incomplete. We introduce *papermage*, an open-source Python toolkit for analyzing and processing visually-rich, structured scientific documents. *papermage* offers clean and intuitive abstractions for seamlessly representing and manipulating both textual and visual document elements. *papermage* achieves this by integrating disparate state-of-the-art NLP and CV models into a unified framework, and provides turn-key recipes for common scientific document processing use-cases. *papermage* has powered multiple research prototypes of AI applications over scientific documents, along with Semantic Scholar's large-scale production system for processing millions of PDFs.

 [github.com/allenai/papermage](https://github.com/allenai/papermage)<sup>1</sup>

## 1 Introduction

Research papers and textbooks are central to the scientific enterprise, and there is increasing interest in developing new tools for extracting knowledge from these visually-rich documents. Recent research has explored, for example, AI-powered reading support for math symbol definitions (Head et al., 2021), in-situ passage explanations or summaries (August et al., 2023; Rachatasumrit et al., 2022; Kim et al., 2023), automatic span highlighting (Chang et al., 2023; Fok et al., 2023b), interactive clipping and synthesis (Kang et al., 2022, 2023)

\*Core contributors; see [author contributions](#) for details.

<sup>1</sup>We use code snippets to illustrate our toolkit's core designs and abstractions. Exact syntax in paper may differ from the actual code, as software will evolve beyond the paper and we opt to simplify syntax when needed for legibility and clarity. We refer readers to our public code for latest documentation.

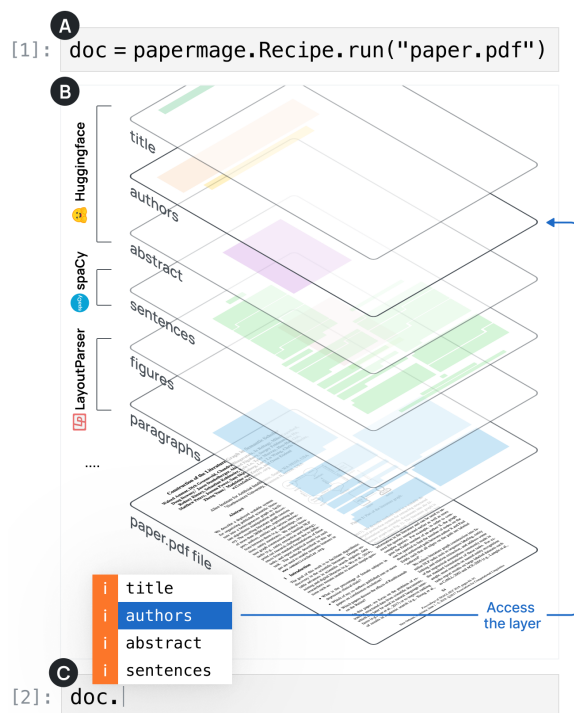


Figure 1: *papermage*'s document creation and representation. (A) Recipes are turn-key methods for processing a PDF. (B) They compose models operating across different data modalities and machine learning frameworks to extract document structure, which we conceptualize as layers of annotation that store textual and visual information. (C) Users can access and manipulate layers.

and more. Further, extracting clean, properly-structured scientific text from PDF documents (Lo et al., 2020; Wang et al., 2020) forms a critical first step in pretraining language models of science (Beltagy et al., 2019; Lee et al., 2019; Gu et al., 2020; Luo et al., 2022; Taylor et al., 2022; Trevartha et al., 2022; Hong et al., 2023), automatic generation of more accessible paper formats (Wang et al., 2021), and developing datasets for scientific natural language processing (NLP) tasks over structured full text (Jain et al., 2020; Subramanian et al., 2020; Dasigi et al., 2021; Lee et al., 2023).

However, this type of NLP research on scientific

corpora is difficult because the documents come in difficult-to-use formats like PDF,<sup>2</sup> and existing tools for working with the documents are limited. Typically, the first step in scientific document processing is to invoke a parser on a document file to convert it into a sequence of tokens and bounding boxes in inferred reading order. Parsers extract only the raw document content, and obtaining richer document structure (e.g., titles, authors, figures) or linguistic structure and semantics (e.g., sentences, discourse units, scientific claims) requires sending the token sequence through downstream models. Unlike more mature parsers (§2.1), these downstream models are often research prototypes (§2.2) that are limited to extracting only a subset of the structures needed for one’s research (e.g., the same model may not provide both sentence splits and figure detection). As a result, users must write extensive custom code that strings pipelines of multiple models together. Research projects using models of different modalities (e.g., combining an image-based formula detector with a text-based definition extractor) can require hundreds of lines of code.

We introduce *papermage*, an open-source Python toolkit for processing scientific documents. Its contributions include (1) *magelib*, a library of primitives and methods for representing and manipulating visually-rich documents as multimodal constructs, (2) *Predictors*, a set of implementations that integrate different state-of-the-art scientific document analysis models into a unified interface, even if individual models are written in different frameworks or operate on different modalities, and (3) *Recipes*, which provide turn-key access to well-tested combinations of individual (often single-modality) modules to form sophisticated, extensible multimodal pipelines.

## 2 Related Work

### 2.1 Turn-key software for scientific documents

Processing visually-rich documents like scientific documents requires a joint understanding of both visual and textual information. In practice, this often requires combining different models into complex processing pipelines. For example, GRO-BID (Grobid, 2008–2023), a widely-adopted software tool for scientific document processing, uses

<sup>2</sup>PDFs store text as character glyphs and their  $(x, y)$  positions on a page. Converting this data to usable text for NLP requires error-prone operations like inferring token boundaries, whitespacing, and reading order using visual positioning.

twelve interdependent sequence labeling models<sup>3</sup> to perform its full text extraction. Other similar tools include CERMINE (Tkaczyk et al., 2015) and ParsCit (Council et al., 2008). While such software is often an ideal choice for off-the-shelf processing, they are not necessarily designed for easy extension and/or integration with newer research models.<sup>4</sup>

### 2.2 Models for scientific document processing

While aforementioned software tools use CRF or BiLSTM-based models, Transformer-based models have seen wide adoption among NLP researchers for their powerful processing capabilities. Recent years have seen the rise of layout-infused Transformers (Xu et al., 2019; Shen et al., 2022; Xu et al., 2021; Huang et al., 2022b; Chen et al., 2023) for processing visually-rich documents, including recovering logical structure (e.g., titles, abstracts) of scientific papers (Huang et al., 2022a). Similarly, computer vision (CV) researchers have also shown impressive capabilities of CNN-based object detection models (Ren et al., 2015; Tan et al., 2020) for segmenting visually-rich documents based on their layout. While these research models are powerful and extensible for research purposes, it often requires significant “glue” code and stitching software tools to create robust processing pipelines. For example, Lincker et al. (2023) bootstraps a sophisticated processing pipeline around a research model for processing children’s textbooks.

### 2.3 Combining models and pipelines

*papermage*’s use case lies between that of turn-key software and a framework for supporting research. Similar to Transformers (Wolfe et al., 2022)’s integration of different research models into standard interfaces, others have done similarly for the visually-rich document domain. *LayoutParser* (Shen et al., 2021) provides models for visually-rich documents and supports the creation of document processing pipelines. *papermage*, in fact, depends on *LayoutParser* for access to vision models, but is designed to also integrate text models which are omitted from

<sup>3</sup><https://grobid.readthedocs.io/en/latest/Training-the-models-of-Grobid/#models>

<sup>4</sup>Most research in NLP requires that a researcher be able to manipulate models within Python. Yet, Grobid requires users to manage a separate service process and send PDFs through a client. In performing evaluation in §3.3, we also found it difficult to run only the model components isolated from PDF utilities, which makes comparison with other research models challenging without significant “glue” code.

```
[3]: doc.sentences[9]
[3]: Entity(id=27, text="in this paper, we...",
          spans=[...], boxes=[...])
```

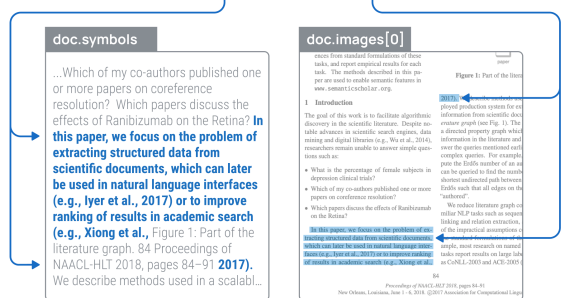


Figure 2: Entities are multimodal content units. Here, spans of a sentence are used to identify its text among all symbols, while boxes map its visual coordinates on a page. spans and boxes can include non-contiguous units, allowing great flexibility in Entities to handle layout nuances. A sentence split across columns/pages and interrupted by floating figures/footnotes would require multiple spans and bounding boxes to represent.

LayoutParser. To allow models of different modalities to work well together, we also developed the `magelib` library (§3.1).

### 3 Design of papermage

papermage is three parts: (1) `magelib`, a library for intuitively representing and manipulating visually-rich documents, (2) Predictors, implementations of models for analyzing scientific papers that unify disparate machine learning frameworks under a common interface, and (3) Recipes, combinations of Predictors that form multimodal pipelines.

#### 3.1 Representing and manipulating visually-rich documents with `magelib`

In this section, we use code snippets to show how our library’s abstractions and syntax are tailored for the visually-rich document problem domain.

**Data Classes.** `magelib` provides three base data classes for representing fundamental elements of visually-rich, structured documents: Document, Layers and Entities. First, a Document might minimally store text as a string of symbols:

```
1 >>> from papermage import Document
2 >>> doc.symbols
3 "Revolt: Collaborative Crowdsourcing..."
```

But visually-rich documents are more than a linearized string. For example, analyzing a scientific paper requires access to its visuospatial layout (e.g.,

pages, blocks, lines), logical structure (e.g., title, abstract, figures, tables, footnotes, sections), semantic units (e.g., paragraphs, sentences, tokens), and more (e.g., citations, terms). In practice, this means different parts of `doc.symbols` can correspond to different paragraphs, sentences, tokens, etc. in the Document, each with its own set of corresponding coordinates representing its visual position on a page.

`magelib` represents structure using Layers that can be accessed as attributes of a Document (e.g., `doc.sentences`, `doc.figures`, `doc.tokens`) (Figure 1). Each Layer is a sequence of content units, called Entities, which store both textual (e.g., spans, strings) and visuospatial (e.g., bounding boxes, pixel arrays) information:

```
1 >>> sentences = Layer(entities=[
2     Entity(...), Entity(...), ...
3 ])
```

See Figure 2 for an example on how “sentences” in a scientific document are represented as Entities. Section §3.2 explains in more detail how a user can generate Entities.

**Methods.** `magelib` also provides a set of functions for building and interacting with data: augmenting a Document with additional Layers, traversing and spatially searching for matching Entities in one Layer, and cross-referencing between Layers (see Figure 3).

A Document that only contains `doc.symbols` can be augmented with additional Layers:

```
1 >>> paragraphs = Layer(...)
2 >>> sentences = Layer(...)
3 >>> tokens = Layer(...)
4
5 >>> doc.add(paragraphs, sentences, tokens)
```

Adding Layers automatically grants users the ability to iterate through Entities and cross-reference intersecting Entities across Layers:

```
1 >>> for paragraph in doc.paragraphs:
2     for sent in paragraph.sentences:
3         for token in sentence.tokens:
4         ...
```

`magelib` also supports cross-modality operations. For example, searching for textual Entities within a visual region on the PDF (See Figure 3 F):

```
1 >>> query = Box(l=423, t=71, w=159, h=87)
2 >>> selection = doc.find(query, "tokens")
3 >>> [t.text for t in selection]
4 ["Techniques", "for", "collecting", ...]
```

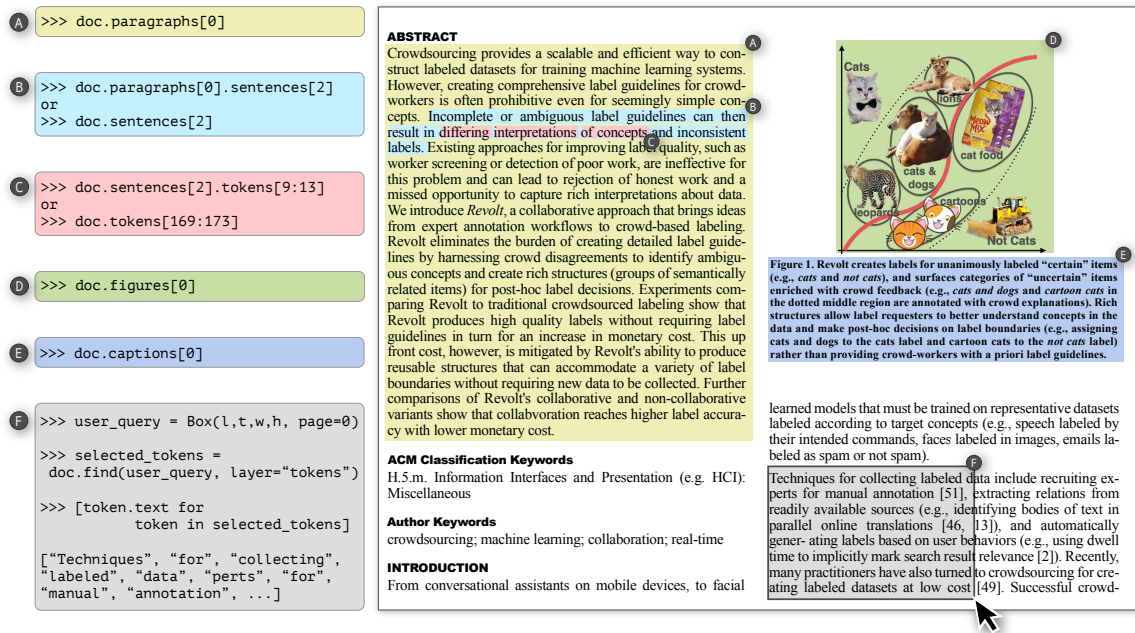


Figure 3: Illustrates how Entities can be accessed flexibly in different ways: (A) Accessing the Entity of the first paragraph in the Document via its own Layer (B) Accessing a sentence via the paragraph Entity or directly via the sentences Layer (C) Similarly, the same tokens can be accessed via the overlapping sentence Entity or directly via the tokens Layer of the Document (where the first tokens are the title of the paper.) (D, E) Figures, captions, tables and keywords can be accessed in similar ways (F) Additionally, given a bounding box (e.g., of a user selected region), papermage can find the corresponding Entities for a given Layer, in this case finding the tokens under the region. Excerpt from Chang et al. (2017).

**Protocols and Utilities.** To instantiate a Document, *magelib* provides protocols and utilities like Parsers and Rasterizers, which hook into off-the-shelf PDF processing tools:<sup>5</sup>

```

1 >>> import papermage as pm
2 >>> parser = pm.PDF2TextParser()
3 >>> doc = parser.parse("...pdf")
4 >>> [token.text for token in doc.tokens]
5 ["Revolt", ":", "Collaborative", ...]
6 >>> doc.images
7 None
8
9 >>> rasterizer = pm.PDF2ImageRasterizer()
10 >>> doc2 = rasterizer.rasterize("...pdf")
11 >>> doc.images = doc2.images
12 >>> doc.images
13 [Image(np.array(...)), ...]

```

In this example, *papermage* runs *PDF2TextParser* (using *pdfplumber*) to extract the textual information from a PDF file. Then it runs *PDF2ImageRasterizer* (using *pdf2image*) to update the first Document with images of pages.

<sup>5</sup>PDFs are not the only way of representing visually-rich documents. For example, many scientific documents are distributed in XML format. As PDFs are the dominant distribution format of scientific documents, we focus our efforts on PDF-specific needs. Nevertheless, we also provide Parsers in *magelib* that can instantiate a Document from XML input. See Appendix A.1.

### 3.2 Interfacing with models for scientific document analysis through Predictors

In §3.1, we described how users create Layers by assembling collections of Entities. But how would they make Entities in the first place?

For example, to identify multimodal structures in visually-rich documents, researchers might want to build complex pipelines that run and combine output from many different models (e.g., computer vision models for extracting figures, NLP models for classifying body text). *papermage* provides a unified interface, called Predictors, to ensure models produce Entities that are compatible with the Document.

*papermage* includes several ready-to-use Predictors that leverage state-of-the-art models to extract specific document structures (Table 1). While *magelib*'s abstractions are general for visually-rich documents, Predictors are optimized for parsing of scientific documents. They are designed to (1) be compatible with models from many different machine learning frameworks, (2) support inference with text-only, vision-only, and multimodal models, and (3) support both adaptation of off-the-shelf, pretrained models as well as

| Use case                | Description  | Examples   |
|-------------------------|--|--|
| Linguistic/<br>Semantic | Segments doc into text units often used for downstream models.   | SentencePredictor wraps sciSpaCy (Neumann et al., 2019) and PySBD (Sadvilkar and Neumann, 2020) to segment sentences. WordPredictor is a custom scikit-learn model to identify broken words split across PDF lines or columns. ParagraphPredictor is a set of heuristics on top of both layout and logical structure models to extract paragraphs.   |
| Layout<br>Structure     | Segments doc into visual block regions.  | BoxPredictor wraps models from LayoutParser (Shen et al., 2021), which provides vision models like EfficientDet (Tan et al., 2020) pretrained on scientific layouts (Zhong et al., 2019).  |
| Logical<br>Structure    | Segments doc into organizational units like title, abstract, body, footnotes, caption, and more.                                       | SpanPredictor wraps Token Classifiers from Transformers (Wolfe et al., 2022), which provides both pretrained weights from VILA (Shen et al., 2022), as well as RoBERTa (Liu et al., 2019), SciBERT (Beltagy et al., 2019) weights that we’ve finetuned on similar data.  |
| Task-<br>specific       | Models for a given scientific document processing task can be used with papermage if wrapped as a Predictor following common patterns. | As many practitioners depend on prompting a model through an API call, we implement APIPredictor which interfaces external APIs, such as GPT-3 (Brown et al., 2020), to perform tasks like question answering over a structured Document. We also implement SnippetRetrievalPredictor which wraps models like Contriever (Izcard et al., 2022) to perform top- <i>k</i> within-document snippet retrieval. See §4 for how these two can be combined. |

Table 1: Types of Predictors implemented in papermage.

| Model                       | Full        |             |             | Grobid Subset |             |             |
|-----------------------------|-------------|-------------|-------------|---------------|-------------|-------------|
|                             | P           | R           | F1          | P             | R           | F1          |
| <i>Grobid<sub>CRF</sub></i> | 40.6        | 38.3        | 39.1        | 81.2          | 76.7        | 78.9        |
| <i>Grobid<sub>NN</sub></i>  | 42.0        | 36.5        | 37.6        | 84.1          | 73.0        | 78.2        |
| <i>RoBERTa</i>              | 75.9        | 80.0        | 76.8        | 82.6          | 83.9        | 83.2        |
| <i>I-VILA</i>               | <b>92.0</b> | <b>94.1</b> | <b>92.7</b> | <b>92.2</b>   | <b>95.2</b> | <b>93.7</b> |

Table 2: Evaluating performance of CoreRecipe for logical structure recovery on S2-VL (Shen et al., 2022). Metrics are computed for token-level classification, macro-averaged over categories. The “Grobid Subset” limits evaluation to only categories for which Grobid returns bounding box information, which was necessary for evaluation on S2-VL. See Appendix A.3 for details.

development of new ones from scratch. Similarly to the Transformers library, a Predictor’s implementation is typically independent from its configuration, allowing users to customize each Predictor by tweaking hyperparameters or loading a different set of weights.

Below, we showcase how a vision model and two text models (both neural and symbolic) can be applied in succession to a single Document. See Table 1 for a summary of supported Predictors.

```

1 >>> import papermage as pm
2 >>> cv = pm.BoxPredictor(...)
3 >>> tables, figures = cv.predict(doc)
4 >>> doc.add(tables, figures)
5
6 >>> nlp_neu = pm.SpanPredictor(...)
7 >>> titles, authors = nlp_neu.predict(doc)
8 >>> doc.add(titles, authors)
9
10 >>> nlp_sym = pm.SentencePredictor(...)
11 >>> sentences = nlp_sym.predict(doc)
12 >>> doc.add(sentences)

```

Predictors return a list of Entities, which can be group\_by() to organize them based on predicted label value (e.g., tokens classified as “title” or “authors”). Finally, these predictions are passed to doc.annotate() to be added to Document.

### 3.3 End-to-end processing with Recipes

Finally, papermage provides predefined combinations of Predictors, called Recipes, for users seeking high-quality options for turn-key processing of visually-rich documents:

```

1 from papermage import CoreRecipe
2 recipe = CoreRecipe()
3 doc = recipe.run("...pdf")
4 doc.captions[0].text
5 >>> "Figure 1. ..."

```

Recipes can also be flexibly modified to support development. For example, our current default combines the pdfplumber PDF parsing utility with the I-VILA (Shen et al., 2022) research model. We show in Table 2 an evaluation comparing this against the same recipe but configured to (1) swap I-VILA for a RoBERTa model, as well as (2) swap both for Grobid API calls. We expect Recipes to appeal to two groups of users—end-to-end consumers, and developers of high-level applications. The former is comprised of developers and researchers who are looking for a one-step solution to multimodal scientific document analysis. The latter are likely developers and researchers looking to combine document structure primitives to build a complex application (see example in §4).

## 4 Vignette: Building an Attributed QA System for Scientific Papers

How could researchers leverage `papermage` for their research? Here, we walk through a user scenario in which a researcher (Lucy) is prototyping an attributed QA system for science.

**System Design.** Drawing inspiration from [Ko et al. \(2020\)](#), [Lee et al. \(2023\)](#), [Fok et al. \(2023a\)](#), and [Newman et al. \(2023\)](#), Lucy is studying how language models can be used to resolve questions that arise while reading a paper (e.g. *What does this mean?* or *What does this refer to?*). In her prototype interface, a user can highlight a passage in a PDF and ask a question about it. A retrieval model then finds relevant passages from the rest of the paper. The prototype then uses the *text* of the retrieved passages along with the user question to prompt a language model to generate an answer. When presenting the answer to the user, the prototype also *visually* highlights the retrieved passages as supporting evidence to the generated answer.

**Getting started quickly.** As a researcher proficient in Python, it only takes Lucy minutes to install `papermage` using `pip` and successfully process a local PDF file by following the example code snippet for `CoreRecipe` in §3.2. In an interactive session, she familiarizes herself with the provided `Layers` by following the traversal, cross-referencing and querying examples in §3.1. She makes sure she can serialize and re-instantiate her `Document` (§A.2).

**Formatting input.** Before using `papermage`, Lucy has prior experience building QA pipelines, but has only dealt with documents as sentence-split text data (e.g., `<List[str]>`). Lucy realizes that she can reuse her prior text-only code with `papermage` by implementing a couple of wrappers to gain additional capabilities: First, she converts a user’s highlighted passage from a visual selection to text following the example in Figure 3F. Next, she converts `Document` to her required text format by following the traversal examples in §3.1 (e.g., using `[s.text for s in doc.sentences]`). Within a few lines of code, Lucy has everything she needs for text-only input to her QA pipeline.

**Formatting output.** Lucy runs her QA system on her newly acquired text data and now has (1) a model-generated answer and (2) several retrieved evidence passages. She realizes that she already has access to the evidences’ bounding boxes via a

similar call to how she defined the model input context (e.g., `[s.bboxes for s in doc.sentences]`). She can easily pass this to the user interface to enable linking to and highlighting of those passages.

**Defining a Predictor.** The pattern Lucy has followed is used in our many `Predictor` implementations: (1) gain access to text by traversing `Layers` (e.g., sentences), (2) perform all usual NLP computation on that text, and (3) format model output as `Entities`. This simple pattern allows users to reuse familiar models in existing frameworks and eschews lengthy onboarding to `papermage`. Lucy wraps her prompting and retrieval code in new classes: `APIPredictor` and `SnippetRetrievalPredictor` (see Table 1).

**Fast iterations.** Leveraging the bounding box data from `papermage` to visually highlight the retrieved passages, Lucy suspects the retrieval component is likely underperforming. She makes a simple edit from `doc.sentences` to `doc.paragraphs` and evaluates system performance under different input granularity. She also realizes the system often retrieves content outside the main body text. She restricts her traversal to filter out paragraphs that overlap with footnotes—`[p.text for p in doc.paragraphs if len(p.footnotes) == 0]`—making clever use of the cross-referencing functionality to detect when a paragraph is actually coming from a footnote. This example demonstrates the versatility of the affordances provided by `magelib`.

## 5 Conclusion

In this work, we’ve introduced `papermage`, an open-source Python toolkit for processing scientific documents. `papermage` was developed to supply high-quality data and reduce friction for research prototype development at [Semantic Scholar](#). Today, it is being used in the production PDF processing pipeline to provide data for both the literature graph ([Ammar et al., 2018](#); [Kinney et al., 2023](#)) and the paper-reading interface ([Lo et al., 2023](#)). It has also been used in working research prototypes which have since contributed to research publications ([Fok et al., 2023b](#); [Kim et al., 2023](#)).<sup>6</sup> We open-source `papermage` in hopes it will simplify research workflows that depend on scientific documents and promote extensions to other visually-rich documents like textbooks ([Lincker et al., 2023](#)) and digitized print media ([Lee et al., 2020](#)).

<sup>6</sup>See a demo of such a prototype [papeo.app/demo](http://papeo.app/demo).



## Ethical Considerations

As a toolkit primarily designed to process scientific documents, there are two areas where papermage could cause harms or have unintended effects.

### Extraction of bibliographic information

papermage could be used to parse author names, affiliation, emails from scientific documents. Like any software, this extraction can be noisy, leading to incorrect parsing and thus mis-attribution of manuscripts. Further, since papermage relies on static PDF documents, rather than metadata dynamically retrieved from publishers, users of papermage need consider how and when extracted names should no longer be associated with authors, a harmful practice called deadnaming (Queer in AI et al., 2023). We recommend papermage users to exercise caution when using our toolkit to extract metadata, to cross-reference extracted content with other sources when possible, and to design systems such that authors have the ability to manually edit any data about themselves.

### Misrepresentation or fabrication of information in documents

In §3, we discussed how papermage can be easily extended to support high-level applications. Such applications might include question answering chatbots, or AI summarizers that perform information synthesis over one or more papermage documents. Such applications typically rely on generative models to produce their output, which might fabricate incorrect information or misstate claims. Developers should be vigilant when integrating papermage output into any downstream application, especially in systems that purport to represent information gathered from scientific publications.

## Acknowledgements

We thank our teammates at Semantic Scholar for their help on this project. In particular: Rodney Kinney provided insight during discussions about how best to represent data extracted from documents; Paul Sayre provided feedback on initial designs of the library; Chloe Anastasiades, Dany Haddad and Egor Klevak tested earlier versions of the library; Tal August, Raymond Fok, and Andrew Head motivated the need for such a toolkit during their internships building augmented reading interfaces; Jaron Lochner and Kelsey MacMillan helped us get additional engineering support; and

Oren Etzioni provided enthusiasm and support for continued investment in this toolkit.

This project was supported in part by NSF Grant OIA-2033558 and NSF Grant CNS-2213656.

## Author Contributions

All authors contributed to the implementation of papermage and/or the writing of this paper.

**Core contributors.** Kyle Lo and Zejiang Shen initiated the project and co-wrote initial implementations of `magelib` and some Predictors. Later, Kyle Lo and Luca Soldaini refactored a majority of `magelib`, Predictors, and added Recipes. Benjamin Newman added new Predictors to support use-cases like those in the Vignette (§4). Joseph Chee Chang implemented an end-to-end web-based visual interface for papermage and helped iterate on papermage’s designs. All core contributors helped with writing. Finally, Kyle Lo led all aspects of the project, including design and implementation, as well as mentorship of other contributors to the toolkit (see below).

**Other contributors.** Russell Authur, Stefan Candra, Yoganand Chandrasekhar, Regan Huff, Amanpreet Singh and Angele Zamarron each worked closely with Kyle Lo to contribute a Predictor to papermage. Erin Bransom and Bailey Kuehl helped with data annotation for training and evaluating those Predictors. Chris Wilhelm provided feedback on papermage’s design and implemented faster indexing of Entities when building Layers. Finally, Marti Hearst, Daniel Weld, and Doug Downey helped with writing and overall advising on the project.

## References

- Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. [Construction of the literature graph in semantic scholar](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.
- Tal August, Lucy Lu Wang, Jonathan Bragg, Marti A. Hearst, Andrew Head, and Kyle Lo. 2023. [Paper](#)

- plain: Making medical research papers approachable to healthcare consumers with natural language processing. *ACM Trans. Comput.-Hum. Interact.*, 30(5).
- 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.
- 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.
- Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. **Revolt: Collaborative crowdsourcing for labeling machine learning datasets**. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 2334–2346, New York, NY, USA. Association for Computing Machinery.
- Joseph Chee Chang, Amy X. Zhang, Jonathan Bragg, Andrew Head, Kyle Lo, Doug Downey, and Daniel S. Weld. 2023. **Citesee: Augmenting citations in scientific papers with persistent and personalized historical context**. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA. Association for Computing Machinery.
- Catherine Chen, Zejiang Shen, Dan Klein, Gabriel Stanovsky, Doug Downey, and Kyle Lo. 2023. **Are layout-infused language models robust to layout distribution shifts? a case study with scientific documents**. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13345–13360, Toronto, Canada. Association for Computational Linguistics.
- Isaac Council, C. Lee Giles, and Min-Yen Kan. 2008. **ParsCit: an open-source CRF reference string parsing package**. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).
- Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. **A dataset of information-seeking questions and answers anchored in research papers**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.
- Raymond Fok, Joseph Chee Chang, Tal August, Amy X. Zhang, and Daniel S. Weld. 2023a. **Qlarify: Bridging scholarly abstracts and papers with recursively expandable summaries**. *arXiv*, abs/2310.07581.
- Raymond Fok, Hita Kambhampettu, Luca Soldaini, Jonathan Bragg, Kyle Lo, Marti Hearst, Andrew Head, and Daniel S Weld. 2023b. **Scim: Intelligent skimming support for scientific papers**. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, page 476–490, New York, NY, USA. Association for Computing Machinery.
- Grobid. 2008–2023. **Grobid**. <https://github.com/kermitt2/grobid>.
- Yu Gu, Robert Tinn, Hao Cheng, Michael R. Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. **Domain-specific language model pretraining for biomedical natural language processing**. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3:1 – 23.
- 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*, CHI '21, New York, NY, USA. Association for Computing Machinery.
- Zhi Hong, Aswathy Ajith, James Pauloski, Eamon Duede, Kyle Chard, and Ian Foster. 2023. **The diminishing returns of masked language models to science**. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1270–1283, Toronto, Canada. Association for Computational Linguistics.
- Po-Wei Huang, Abhinav Ramesh Kashyap, Yanxia Qin, Yajing Yang, and Min-Yen Kan. 2022a. **Lightweight contextual logical structure recovery**. In *Proceedings of the Third Workshop on Scholarly Document Processing*, pages 37–48, Gyeongju, Republic of Korea. Association for Computational Linguistics.
- Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022b. **Layoutlmv3: Pre-training for document ai with unified text and image masking**. *Proceedings of the 30th ACM International Conference on Multimedia*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. **Unsupervised dense information retrieval with contrastive learning**. *Transactions on Machine Learning Research*.
- Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. **SciREX: A challenge dataset for document-level information extraction**. In

- Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.
- Hyeonsu B. Kang, Joseph Chee Chang, Yongsung Kim, and Aniket Kittur. 2022. [Threddy: An interactive system for personalized thread-based exploration and organization of scientific literature](#). In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, New York, NY, USA. Association for Computing Machinery.
- Hyeonsu B. Kang, Sherry Tongshuang Wu, Joseph Chee Chang, and Aniket Kittur. 2023. [Synergi: A mixed-initiative system for scholarly synthesis and sense-making](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery.
- Tae Soo Kim, Matt Latzke, Jonathan Bragg, Amy X. Zhang, and Joseph Chee Chang. 2023. [Papeos: Augmenting research papers with talk videos](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.
- Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld. 2023. [The Semantic Scholar Open Data Platform](#). *ArXiv*, abs/2301.10140.
- Wei-Jen Ko, Te-yuan Chen, Yiyan Huang, Greg Durrett, and Junyi Jessy Li. 2020. [Inquisitive question generation for high level text comprehension](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6544–6555, Online. Association for Computational Linguistics.
- Benjamin Charles Germain Lee, Jaime Mears, Eileen Jakeway, Meghan Ferriter, Chris Adams, Nathan Yarasavage, Deborah Thomas, Kate Zwaard, and Daniel S. Weld. 2020. [The newspaper navigator dataset: Extracting headlines and visual content from 16 million historic newspaper pages in chronicling america](#). In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20*, page 3055–3062, New York, NY, USA. Association for Computing Machinery.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. [BioBERT: a pre-trained biomedical language representation model for biomedical text mining](#). *Bioinformatics*, 36(4):1234–1240.
- Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong-In Lee, and Moontae Lee. 2023. [QASA: Advanced question answering on scientific articles](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19036–19052. PMLR.
- Élise Lincker, Olivier Pons, Camille Guinaudeau, Isabelle Barbet, Jérôme Dupire, Céline Hudelot, Vincent Mousseau, and Caroline Huron. 2023. [Layout and activity-based textbook modeling for automatic pdf textbook extraction](#). In *iTextbooks@AIED*.
- 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*, abs/1907.11692.
- Kyle Lo, Joseph Chee Chang, Andrew Head, Jonathan Bragg, Amy X. Zhang, Cassidy Trier, Chloe Anastasiades, Tal August, Russell Authur, Danielle Bragg, Erin Bransom, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Yen-Sung Chen, Evie Yu-Yen Cheng, Yvonne Chou, Doug Downey, Rob Evans, Raymond Fok, Fangzhou Hu, Regan Huff, Dongyeop Kang, Tae Soo Kim, Rodney Kinney, Aniket Kittur, Hyeonsu Kang, Egor Klevak, Bailey Kuehl, Michael Langan, Matt Latzke, Jaron Lochner, Kelsey MacMillan, Eric Marsh, Tyler Murray, Aakanksha Naik, Ngoc-Uyen Nguyen, Srishti Palani, Soya Park, Caroline Paulic, Napol Rachatasumrit, Smita Rao, Paul Sayre, Zejiang Shen, Pao Siangliulue, Luca Soldaini, Huy Tran, Madeleine van Zuylen, Lucy Lu Wang, Christopher Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Marti A. Hearst, and Daniel S. Weld. 2023. [The Semantic Reader Project: Augmenting Scholarly Documents through AI-Powered Interactive Reading Interfaces](#). *ArXiv*, abs/2303.14334.
- 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.
- Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. [Biogpt: Generative pre-trained transformer for biomedical text generation and mining](#). *Briefings in bioinformatics*.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. [ScispaCy: Fast and robust models for biomedical natural language processing](#). In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

- Benjamin Newman, Luca Soldaini, Raymond Fok, Arman Cohan, and Kyle Lo. 2023. A question answering framework for decontextualizing user-facing snippets from scientific documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Organizers of Queer in AI, Anaelia Ovalle, Arjun Subramonian, Ashwin Singh, Claas Voelcker, Danica J. Sutherland, Davide Locatelli, Eva Breznik, Filip Klubicka, Hang Yuan, Hetvi J, Huan Zhang, Jaidev Shriram, Kruno Lehman, Luca Soldaini, Maarten Sap, Marc Peter Deisenroth, Maria Leonor Pacheco, Maria Ryskina, Martin Mundt, Milind Agarwal, Nyx Mclean, Pan Xu, A Pranav, Raj Korpan, Ruchira Ray, Sarah Mathew, Sarthak Arora, St John, Tanvi Anand, Vishakha Agrawal, William Agnew, Yanan Long, Zijie J. Wang, Zeerak Talat, Avijit Ghosh, Nathaniel Dennler, Michael Noseworthy, Sharvani Jha, Emi Baylor, Aditya Joshi, Natalia Y. Bilenko, Andrew Mcnamara, Raphael Gontijo-Lopes, Alex Markham, Evyn Dong, Jackie Kay, Manu Saraswat, Nikhil Vytla, and Luke Stark. 2023. [Queer In AI: A Case Study in Community-Led Participatory AI](#). In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, FAccT '23*, page 1882–1895, New York, NY, USA. Association for Computing Machinery.
- Napol Rachatasumrit, Jonathan Bragg, Amy X. Zhang, and Daniel S Weld. 2022. [Citeread: Integrating localized citation contexts into scientific paper reading](#). In *27th International Conference on Intelligent User Interfaces, IUI '22*, page 707–719, New York, NY, USA. Association for Computing Machinery.
- Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. [Faster r-cnn: Towards real-time object detection with region proposal networks](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149.
- Nipun Sadvilkar and Mark Neumann. 2020. [PySBD: Pragmatic sentence boundary disambiguation](#). In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 110–114, Online. Association for Computational Linguistics.
- Zejiang Shen, Kyle Lo, Lucy Lu Wang, Bailey Kuehl, Daniel S. Weld, and Doug Downey. 2022. [VILA: Improving structured content extraction from scientific PDFs using visual layout groups](#). *Transactions of the Association for Computational Linguistics*, 10:376–392.
- Zejiang Shen, Ruochen Zhang, Melissa Dell, B. Lee, Jacob Carlson, and Weining Li. 2021. [Layoutparser: A unified toolkit for deep learning based document image analysis](#). In *IEEE International Conference on Document Analysis and Recognition*.
- Sanjay Subramanian, Lucy Lu Wang, Ben Bogin, Sachin Mehta, Madeleine van Zuylen, Sravanthi Parasa, Sameer Singh, Matt Gardner, and Hannaneh Hajishirzi. 2020. [MedICaT: A dataset of medical images, captions, and textual references](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2112–2120, Online. Association for Computational Linguistics.
- M. Tan, R. Pang, and Q. V. Le. 2020. [Efficientdet: Scalable and efficient object detection](#). In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, Los Alamitos, CA, USA. IEEE Computer Society.
- Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony S. Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. [Galactica: A large language model for science](#). *ArXiv*, abs/2211.09085.
- pdf2image. 2023. [pdf2image](#). <https://github.com/Belval/pdf2image>.
- pdfplumber. 2023. [pdfplumber](#). <https://github.com/jsvine/pdfplumber>.
- Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Lukasz Bolikowski. 2015. [Cermine: Automatic extraction of structured metadata from scientific literature](#). *Int. J. Doc. Anal. Recognit.*, 18(4):317–335.
- Amalie Trewartha, Nicholas Walker, Haoyan Huo, Sanghoon Lee, Kevin Cruse, John Dagdelen, Alex Dunn, Kristin Aslaug Persson, Gerbrand Ceder, and Anubhav Jain. 2022. [Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science](#). *Patterns*, 3.
- Lucy Lu Wang, Isabel Cachola, Jonathan Bragg, Evie (Yu-Yen) Cheng, Chelsea Hess Haupt, Matt Latzke, Bailey Kuehl, Madeleine van Zuylen, Linda M. Wagner, and Daniel S. Weld. 2021. Improving the accessibility of scientific documents: Current state, user needs, and a system solution to enhance scientific pdf accessibility for blind and low vision users. *ArXiv*, abs/2105.00076.
- Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. [CORD-19: The COVID-19 open research dataset](#). In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.
- Rosalee Wolfe, John McDonald, Ronan Johnson, Ben Sturr, Syd Klinghoffer, Anthony Bonzani, Andrew Alexander, and Nicole Barnekow. 2022. [Supporting mouthing in signed languages: New innovations and a proposal for future corpus building](#). In *Proceedings of the 7th International Workshop on Sign Language*

*Translation and Avatar Technology: The Junction of the Visual and the Textual: Challenges and Perspectives*, pages 125–130, Marseille, France. European Language Resources Association.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. [LayoutLMv2: Multi-modal pre-training for visually-rich document understanding](#). 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 2579–2591, Online. Association for Computational Linguistics.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2019. [Layoutlm: Pre-training of text and layout for document image understanding](#). *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. [Publaynet: largest dataset ever for document layout analysis](#). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE.

## A Appendix

### A.1 Comparison and Compatibility with XML

One can view Layers as capturing content hierarchy (e.g., tokens vs sentences) similar to that of other structured document representations, like TEI XML trees. We note that Layers are stored as unordered attributes and don't require nesting. This allows for specific cross-layer referencing operations that don't adhere to strict nesting relationships. For example:

```
1 for sentence in doc.sentences:
2     for line in sentence.lines:
3         ...
```

Recall that a sentence can begin or end midway through a line and cross multiple lines (§3.1). Similarly, not all lines are exactly contained within the boundaries of a sentence. As such, sentences and lines are not strictly nested within each other. This relationship would be difficult to encode in an XML format adhering to document tree structure.

Regardless, the way we represent structure in documents is highly versatile. We demonstrate this by also implementing GrobidParser as an alternative to the PDF2TextParser in §3.1. GrobidParser invokes Grobid to process PDFs, and reads the resulting TEI XML file generated by Grobid by converting each XML tag of a common level into an Entity of its own Layer. We use this to perform the evaluation in Table 2.

### A.2 Additional magelib Protocols and Utilities

**Serialization.** Any Document and all of its Layers can be exported to a JSON format, and perfectly reconstructed:

```
1 import json
2 with open("...json", "w") as f_out:
3     json.dump(doc.to_json(), f_out)
4
5 with open("...json", "r") as f_in:
6     doc = json.load(f_in)
```

### A.3 Evaluating papermage's CoreRecipe against Grobid

Here, we detail how we performed the evaluation reported in §3.3 (Table 2). We also provide a full breakdown by category in Table 3.

As described earlier in the paper, Grobid is quite difficult to evaluate as it is developed with tight coupling between the PDF parser (pdfalto) and

the models it employs to perform logical structure recovery over the resulting token stream. As such, there is no straightforward way to run just the model components of Grobid on an alternative token stream like that provided in the S2-VL (Shen et al., 2022) dataset.

To perform this baseline evaluation, we ran the original PDFs that were annotated for S2-VL through our GrobidParser using v0.7.3. Grobid also returns bounding boxes of some predicted categories (e.g., authors, abstract, paragraphs). We use these bounding boxes to create Entities that we annotate on a Document constructed manually from S2-VL data. Using magelib cross-layer referencing, we were able to match Grobid predictions to S2-VL data to perform this evaluation.

Though we found there are certain categories for which bounding box information was either not available (e.g., Titles) or Grobid simply did not return that output (e.g., Figure text extraction). These are represented by zeros in Table 3, which contributes to the lower scores in Table 2 after macro averaging. For a more apples-to-apples comparison, we also included a "Grobid Subset" evaluation which restricted to just categories in S2-VL for which Grobid produced bounding box information.

In addition to Grobid, we evaluate two of our provided Transformer-based models. The RoBERTa-large (Liu et al., 2019) model is a Transformers token classification model that we finetuned on the S2-VL training set. The I-VILA model is a layout-infused Transformer model pretrained by Shen et al. (2022) on the S2-VL training set. Like we did with Grobid, we ran our CoreRecipe using these two models on the original PDFs in S2-VL, and performed a similar token mapping operation since our PDF2TextParser also produces a different token stream than that provided in S2-VL.

At the end of the day, the Transformer-based models performed better at this task than Grobid. This is unsurprising given expected improvements using a Transformer model over a CRF or BiLSTM. The Transformer models were also trained on S2-VL data, which gave them an advantage over Grobid. Overall, this evaluation intended to show how papermage enables cross-system comparisons, even eschewing token stream incompatibility, and to illustrate an upper bound of the performance left on the table by existing software systems that don't use of state-of-the-art models.

| Structure Category                  | GROBID <sub>CRF</sub> |      |      | GROBID <sub>NN</sub> |      |      | RoBERTa |      |      | I-VILA |      |      |
|-------------------------------------|-----------------------|------|------|----------------------|------|------|---------|------|------|--------|------|------|
|                                     | P                     | R    | F1   | P                    | R    | F1   | P       | R    | F1   | P      | R    | F1   |
| Abstract                            | 81.9                  | 89.1 | 85.3 | 85.3                 | 89.8 | 87.5 | 89.2    | 93.7 | 91.4 | 97.4   | 98.3 | 97.8 |
| Author                              | 55.2                  | 42.6 | 48.1 | 75.1                 | 14.0 | 23.6 | 87.5    | 73.5 | 79.9 | 65.5   | 96.9 | 78.2 |
| Bibliography                        | 96.5                  | 98.6 | 97.5 | 95.5                 | 97.6 | 96.5 | 93.6    | 93.3 | 93.5 | 99.7   | 98.2 | 99.0 |
| Caption                             | 70.3                  | 70.0 | 70.2 | 70.2                 | 69.7 | 70.0 | 80.0    | 77.3 | 78.6 | 93.1   | 89.6 | 91.3 |
| Equation                            | 71.1                  | 85.3 | 77.6 | 71.1                 | 85.3 | 77.6 | 55.0    | 85.7 | 67.0 | 90.7   | 94.2 | 92.4 |
| Figure                              | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 88.9    | 82.3 | 85.4 | 99.8   | 96.8 | 98.3 |
| Footer                              | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 56.1    | 59.9 | 57.9 | 96.8   | 78.1 | 86.5 |
| Footnote                            | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 59.8    | 44.3 | 50.9 | 80.2   | 93.5 | 86.3 |
| Header                              | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 40.5    | 84.3 | 54.7 | 92.9   | 99.1 | 95.9 |
| Keywords                            | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 93.8    | 97.1 | 95.4 | 96.9   | 99.4 | 98.1 |
| List                                | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 61.9    | 63.8 | 62.9 | 76.7   | 82.4 | 79.4 |
| Paragraph                           | 94.5                  | 89.8 | 92.1 | 94.4                 | 89.9 | 92.1 | 93.5    | 93.0 | 93.3 | 98.7   | 97.9 | 98.3 |
| Section                             | 83.0                  | 79.4 | 81.1 | 83.0                 | 79.4 | 81.1 | 67.7    | 82.7 | 74.4 | 96.2   | 91.6 | 93.9 |
| Table                               | 97.3                  | 58.6 | 73.2 | 97.9                 | 58.6 | 73.3 | 94.7    | 71.8 | 81.7 | 96.1   | 94.9 | 95.5 |
| Title                               | 0.0                   | 0.0  | 0.0  | 0.0                  | 0.0  | 0.0  | 76.3    | 96.7 | 85.3 | 98.7   | 99.9 | 99.3 |
| <b>Macro Avg</b><br>(Full S2-VL)    | 40.6                  | 38.3 | 39.1 | 42.0                 | 36.5 | 37.6 | 75.9    | 80.0 | 76.8 | 92.0   | 94.1 | 92.7 |
| <b>Macro Avg</b><br>(Grobid Subset) | 81.2                  | 76.7 | 78.9 | 84.1                 | 73.0 | 78.2 | 82.6    | 83.9 | 83.2 | 92.2   | 95.2 | 93.7 |

Table 3: Evaluating CoreRecipe for logical structure recovery on S2-VL (Shen et al., 2022). These are per-category metrics for Table 2. Metrics are computed for token-level classification, macro-averaged over categories. The “Grobid Subset” limits evaluation to only categories for which Grobid returns bounding box information, which was necessary for evaluation on S2-VL.

# OmniEvent: A Comprehensive, Fair, and Easy-to-Use Toolkit for Event Understanding

Hao Peng<sup>1\*</sup>, Xiaozhi Wang<sup>1\*</sup>, Feng Yao<sup>3</sup>, Zimu Wang<sup>4</sup>,  
Chuzhao Zhu<sup>1</sup>, Kaisheng Zeng<sup>1</sup>, Lei Hou<sup>1,2†</sup>, Juanzi Li<sup>1,2</sup>

<sup>1</sup>Department of Computer Science and Technology, BNRist;

<sup>2</sup>KIRC, Institute for Artificial Intelligence,

Tsinghua University, Beijing, 100084, China

<sup>3</sup>University of California, San Diego <sup>4</sup>Xi'an Jiaotong-Liverpool University

{peng-h21, wangxz20}@mails.tsinghua.edu.cn

## Abstract

Event understanding aims at understanding the content and relationship of events within texts, which covers multiple complicated information extraction tasks: event detection, event argument extraction, and event relation extraction. To facilitate related research and application, we present an event understanding toolkit OmniEvent, which features three desiderata: (1) **Comprehensive**. OmniEvent supports mainstream modeling paradigms of all the event understanding tasks and the processing of 15 widely-used English and Chinese datasets. (2) **Fair**. OmniEvent carefully handles the inconspicuous evaluation pitfalls reported in Peng et al. (2023), which ensures fair comparisons between different models. (3) **Easy-to-use**. OmniEvent is designed to be easily used by users with varying needs. We provide off-the-shelf models that can be directly deployed as web services. The modular framework also enables users to easily implement and evaluate new event understanding models with OmniEvent. The toolkit<sup>1</sup> is publicly released along with the demonstration website and video<sup>2</sup>.

## 1 Introduction

Correctly understanding events is fundamental for humans to understand the world. Event understanding requires identifying real-world events mentioned in texts and analyzing their relationships, which naturally benefits various downstream applications, such as stock prediction (Ding et al., 2015), adverse drug event detection (Wunnava et al., 2019), narrative event prediction (Wang et al., 2021a), and legal case analysis (Yao et al., 2022).

As illustrated in Figure 1, event understanding covers three complicated information extraction tasks: (1) event detection (ED), which is to detect

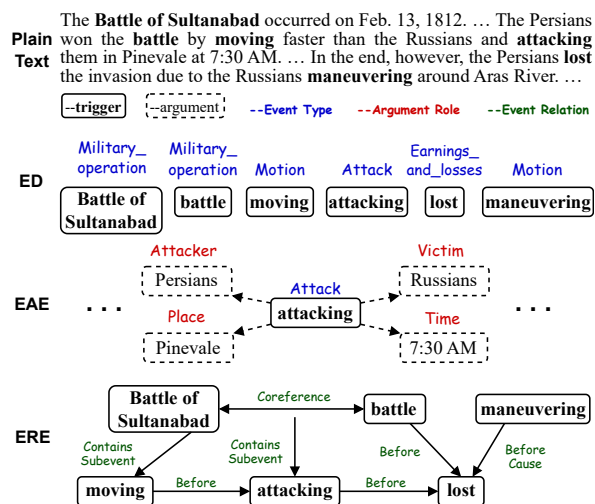


Figure 1: An illustration for the event understanding tasks, including event detection (ED), event argument extraction (EAE), and event relation extraction (ERE).

the event triggers (keywords or phrases evoking events in texts) and classify their event types, (2) event argument extraction (EAE), which is to extract the event arguments for each trigger and classify their argument roles, and (3) event relation extraction (ERE), which is to identify the complex relationships between events, typically including temporal, causal, coreference, and subevent relations. ED and EAE together constitute the conventional event extraction (EE) task.

In recent years, event understanding research has grown rapidly (Ma et al., 2022; Wang et al., 2022; Yue et al., 2023; Huang et al., 2023), and multiple practical systems (Wadden et al., 2019; Lin et al., 2020; Zhang et al., 2020; Du et al., 2022; Zhang et al., 2022) have been developed. However, as shown in Table 1, existing systems exhibit several non-negligible issues: (1) **Incomprehensive Tasks**. Existing systems mainly focus on the two EE subtasks and rarely cover the whole event understanding pipeline with ERE tasks. The notable exception EventPlus (Ma et al., 2021) merely cov-

\* Equal contribution.

† Corresponding author: L.Hou

<sup>1</sup><https://github.com/THU-KEG/OmniEvent>

<sup>2</sup><https://omnievent.xlore.cn/>



ers the temporal relations. (2) **Limited Support for Redevelopment and Evaluation.** Most of the existing event understanding systems are highly integrated and not extensible, which means users cannot easily develop new models within their frameworks. Especially considering the recent rise of large language models (LLMs)<sup>3</sup>, adequate support for LLMs is urgent but often missing. Moreover, the complicated data processing and evaluation details often lead to inconsistent and unfair evaluation results (Peng et al., 2023), but existing systems do not pay much attention to evaluations.

To address these issues, we develop OmniEvent, a comprehensive, fair, and easy-to-use toolkit for event understanding, which has three main features: (1) **Comprehensive Support for Task, Model, and Dataset.** OmniEvent supports end-to-end event understanding from plain texts, i.e., all the ED, EAE, and ERE tasks. For ED and EAE, we classify the mainstream methods into four paradigms, including classification, sequence labeling, span prediction, and conditional generation. We implement various representative methods for each paradigm. For ERE, we provide a unified modeling framework and implement a basic pairwise classification method (Wang et al., 2022). We also cover the preprocessing of 15 widely-used English and Chinese datasets. (2) **Fair Evaluation.** As found in Peng et al. (2023), there are three major pitfalls hidden in EE evaluation, including data processing discrepancy, output space discrepancy, and absence of pipeline evaluation. OmniEvent implements all the proposed remedies to help users avoid them. Specifically, we implement unified pre-processing for all the datasets and a method to convert the predictions of different paradigms into a unified space. OmniEvent also provides unified prediction triggers of supported datasets for fair pipeline comparisons. (3) **Easy-to-Use for Various Needs.** We design a modular and extensible framework for OmniEvent, which appeals to users with various needs. We provide several off-the-shelf models that can be easily deployed and used by users interested in applications. Model developers and researchers can train implemented methods within several lines of code or customize their own models and evaluate them. By integrating Transformers (Wolf et al., 2020) and DeepSpeed (Rasley et al., 2020), OmniEvent also supports efficiently

<sup>3</sup>The definition of LLM is vague. Here we use “LLM” to refer to models with more than 10 billion parameters.

| System    | EE | ERE | #Supported Models | #Supported Datasets | LLM Support |
|-----------|----|-----|-------------------|---------------------|-------------|
| DYGIE     | ✓  | ✗   | 1                 | 1                   | ✗           |
| OneIE     | ✓  | ✗   | 1                 | 4                   | ✗           |
| OpenUE    | ✓  | ✗   | 1                 | 2                   | ✗           |
| EventPlus | ✓  | ✓   | 1                 | N/A                 | ✗           |
| FourIE    | ✓  | ✗   | 1                 | N/A                 | ✗           |
| RESIN-11  | ✓  | ✗   | 1                 | N/A                 | ✗           |
| DeepKE    | ✓  | ✗   | 2                 | 1                   | ✓           |
| OmniEvent | ✓  | ✓   | >20               | 15                  | ✓           |

Table 1: Comparisons between OmniEvent and other event understanding systems. The number of supported models and datasets only includes those of event understanding tasks. N/A denotes that the system is an integrated service and does not process benchmark datasets. For OmniEvent, the module combination enables many possible models and 20 is the number of models we have tested for usability.

fine-tuning LLMs as backbones.

To demonstrate the effectiveness of OmniEvent, we present the results of several implemented methods on widely-used benchmarks. We also conduct experiments with models at different scales and show that fine-tuning LLMs helps achieve better event understanding results. We hope OmniEvent could facilitate the research and applications of event understanding.

## 2 Related Work

With the advancement of research in NLP, various toolkits or systems for event understanding have been developed. They tend to focus on developing advanced EE systems to achieve improved results on public benchmarks (Wadden et al., 2019; Lin et al., 2020; Nguyen et al., 2021) or perform robustly in real-world scenarios (Vossen et al., 2016; Du et al., 2022). However, these toolkits or systems, designed based on a specific EE model, do not support comprehensive implementations of EE models and are inconvenient for secondary development. There is also some work that has meticulously designed user-friendly algorithmic frameworks (Zhang et al., 2020, 2022), which are convenient for usage and secondary development. However, they are not specifically designed for event understanding, hence the corresponding support is limited. EventPlus (Ma et al., 2021) is the only work supporting the entire event understanding pipeline but it only supports temporal relation extraction and does not provide comprehensive implementations of event understanding models. Moreover, existing work also neglects the discrepan-

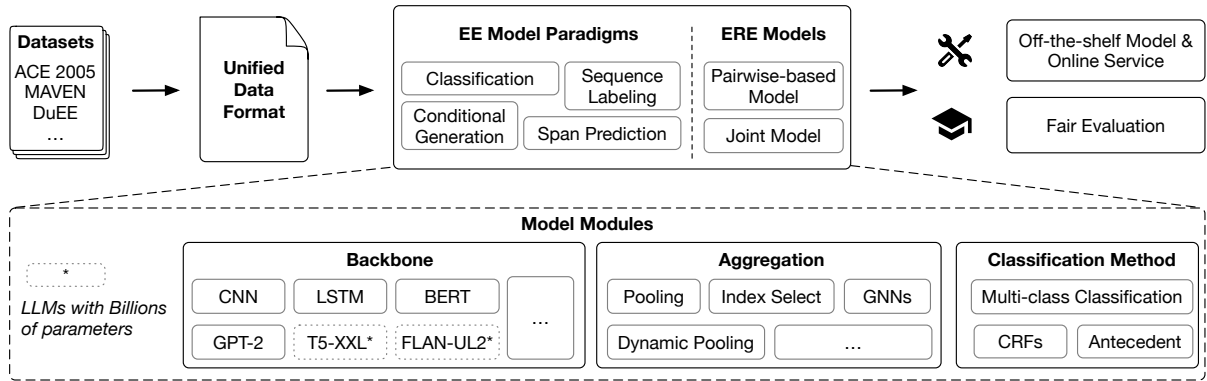


Figure 2: Overview of the OmniEvent toolkit. OmniEvent can serve as a system offering event understanding services to users, while also serving as a toolkit for researchers in model development and evaluation. OmniEvent provides pre-processing scripts for widely-used datasets and converts the datasets into a unified data format. OmniEvent provides modular components and users can easily develop a new model based on the components. OmniEvent also supports large language models (T5-XXL (Raffel et al., 2020) and FLAN-UL2 (Tay et al., 2023)).

cies in EE evaluation as mentioned in Peng et al. (2023), which may result in unfair comparison. Finally, in the era of LLMs, existing work (except for DeepKE) also lacks support for LLMs.

Considering the mentioned issues, we present OmniEvent, a comprehensive, fair, and easy-to-use toolkit for event understanding. Compared to other systems in Table 1, OmniEvent supports the entire event understanding pipeline and comprehensively implements various models. OmniEvent also supports efficient fine-tuning and inference of LLMs. Meanwhile, OmniEvent provides respective remedies for eliminating the discrepancies as mentioned in Peng et al. (2023). With a modular implementation and several released off-the-shelf models, OmniEvent is user-friendly and easy to use.

### 3 The OmniEvent Toolkit

We introduce the overview (§ 3.1) and main features of OmniEvent (§§ 3.2 to 3.4), as well as an online demonstration (§ 3.5) powered by OmniEvent.

#### 3.1 Overview

The overall architecture of OmniEvent is illustrated in Figure 2. OmniEvent provides a data pre-processing module for unified pre-processing. Users can either use the supported datasets or customize their own datasets. After pre-processing, OmniEvent provides a flexible modular framework for model implementation. OmniEvent abstracts and disassembles the mainstream models into three basic modules and implements the basic modules in a highly encapsulated way. By combining our provided modules or implementing their own modules,

users can easily assemble a model. OmniEvent reproduces several widely-used models in this way. Finally, OmniEvent provides a fair evaluation protocol to convert predictions of different models into a unified and comparable output space.

#### 3.2 Comprehensive Support

OmniEvent implements the entire event understanding pipeline, i.e., all the ED, EAE, and ERE tasks, and can serve as a one-stop event understanding platform. Furthermore, OmniEvent provides comprehensive coverage of models and datasets.

**Models** OmniEvent comprehensively implements representative models for ED, EAE, and ERE. For ED and EAE, OmniEvent covers four mainstream method paradigms, which contain: (1) classification methods, including DMCNN (Chen et al., 2015), DMBERT (Wang et al., 2019), and CLEVE (Wang et al., 2021b), which classify event or argument candidates into appropriate types, (2) sequence labeling methods, including BiLSTM+CRF (Wang et al., 2020b) and BERT+CRF (Wang et al., 2020b), which labels the sequences with the BIO format, (3) span prediction method, including EEQA (Du and Cardie, 2020), which predicts the boundaries of event and argument spans, (4) conditional generation method, including Text2Event (Lu et al., 2021), which directly generates the answers. Moreover, as shown in Figure 2, OmniEvent implements various basic modules and the users can easily combine different modules to build new models, e.g., combining GPT-2 (Radford et al., 2019) and CRF (Lafferty et al., 2001) (GPT-2+CRF). For event relation ex-

|     |  |
|-----|--|
| EE  | ACE 2005 (Walker et al., 2006), TAC KBP (Ellis et al., 2014, 2015, 2016; Getman et al., 2017), RichERE (Song et al., 2015), MAVEN (Wang et al., 2020b), ACE 2005 (zh) (Walker et al., 2006), LEVEN (Yao et al., 2022), DuEE (Li et al., 2020), FewFC (Zhou et al., 2021) |
| ERE | MAVEN-ERE (Wang et al., 2022), ACE 2005 (Walker et al., 2006), TB-Dense (Chambers et al., 2014), MATRES (Ning et al., 2018b), TCR (Ning et al., 2018a), CausalTB (Mirza et al., 2014), EventStoryLine (Caselli and Vossen, 2017), HiEve (Glavaš et al., 2014)            |

Table 2: Currently supported datasets in OmniEvent. *Italics* represent Chinese datasets.

```

from OmniEvent import convert_SL, convert_SP, convert_CG

text = "City A suffers a terrorist attack in 2021 ."
tokens = text.split()
events = [{
    "type": "attack",
    "trigger": "terrorist attack",
    "offset": [4, 6]
}]

# predictions generated by users
predictions_SL = [0, 0, 0, B-Attack, I-Attack, I-Attack, 0,
0, 0]
# obtain comparable results
results = convert_SL([predictions_SL], [events], [tokens])

predictions_SP = [{"offset": [3, 6], "type": "attack"}]
results = convert_SP([predictions_SP], [events], [tokens])

# without offsets
predictions_CG = [{"trigger": "a terrorist attack", "type":
"attack"}]
results = convert_CG([predictions_CG], [events], [tokens])

```

Code 1: Example for converting the sequence labeling, span prediction, and conditional generation predictions into a unified output space.

traction, OmniEvent implements a unified pairwise relation extraction framework. Especially for the event coreference resolution task, OmniEvent develops an antecedent ranking method. As extracting different relations (causal, temporal) may benefit each other (Wang et al., 2022), we develop a joint event relation extraction model in OmniEvent.

**Datasets** As shown in Table 2, OmniEvent includes various widely-used Chinese and English event understanding datasets, covering general, legal, and financial domains. For each included dataset, we provide a pre-processing script to convert the dataset into a unified format, as shown in appendix A. For datasets with different pre-processing scripts, e.g., ACE 2005, OmniEvent provides all the mainstream scripts for users.

### 3.3 Fair Evaluation

As discussed in Peng et al. (2023), there exist

```

from OmniEvent.infer import infer
# input text
text = "U.S. and British troops were moving on the strategic
southern port city of Basra Saturday after a massive
aerial assault pounded Baghdad at dawn"
# event detection
ed_results = infer(text=text, task="ED")
# end-to-end event extraction
ee_results = infer(text=text, task="EE")
# end-to-end event understanding
# event extraction & relation extraction
all_results = infer(text=text, task="EE & ERE")

```

Code 2: Example of using inference interface and off-the-shelf models for event understanding.

several pitfalls in EE evaluation that significantly influence the fair comparison of different models. They are in three aspects: data-preprocessing discrepancy, output space discrepancy, and absence of pipeline evaluation. OmniEvent proposes remedies for eliminating them.

**Specify data pre-processing** As the data pre-processing discrepancy mainly comes from using different processing options, OmniEvent provides all the widely-used data pre-processing scripts. Users only need to specify the pre-processing script for comparable results with previous studies.

**Standardize output space** As suggested in Peng et al. (2023), OmniEvent provides several easy-to-use functions to convert the predictions of different models into a unified output space. Code 1 shows the conversion codes of sequence labeling, span prediction, and conditional generation predictions for event detection. Users can easily utilize the functions to obtain fair and comparable results.

**Pipeline evaluation** The pipeline evaluation requires conducting EAE based on predicted triggers. Therefore, the results of EAE models are comparable only when using the same predicted triggers. OmniEvent provides a unified set of predicted triggers for widely-used datasets. Specifically, OmniEvent leverages CLEVE (Wang et al., 2021b), an advanced ED model, to predict triggers for widely-used EE datasets: ACE 2005, KBP 2016, KBP 2017, and RichERE.

### 3.4 Easy-to-Use

OmniEvent is designed to be user-friendly and easy to use. Specifically, OmniEvent incorporates the following designs.

**Easy start with off-the-shelf models** OmniEvent provides several off-the-shelf models for

event understanding. Specifically, we train a multilingual T5 (Xue et al., 2021) for ED and EAE on the collection of included EE datasets, respectively. And we train a joint ERE model based on RoBERTa (Liu et al., 2019) on the training set of MAVEN-ERE. As shown in Code 2, OmniEvent provides an interface for inference and users can easily use these models in their applications with a few lines of code.

**Modular implementation** As shown in Figure 2, OmniEvent abstracts and disassembles the mainstream models into basic modules. The backbone module implements various text encoders, such as CNN (Krizhevsky et al., 2012) and BERT (Devlin et al., 2019), to encode plain texts into low-dimension dense vectors. The backbone module also supports LLMs such as T5-XXL (Raffel et al., 2020) and FLAN-UL2 (Tay et al., 2023). The aggregation module includes various aggregation operations, which aggregate and convert the dense vectors into representations of events, arguments, and relations. The classification module projects the representations into distributions of classification candidates. With the highly modular implementation, users can easily combine the basic modular components to develop new models.

**Efficient support for LLMs** OmniEvent is built upon Huggingface’s Transformers (Wolf et al., 2020) and DeepSpeed (Rasley et al., 2020), an efficient deep learning optimization library. With the built-in DeepSpeed support, OmniEvent can be used to train and infer LLMs efficiently with only modifications of the startup shell scripts.

### 3.5 Online Demonstration

Besides the OmniEvent toolkit, we also develop an online demonstration system<sup>4</sup> powered by OmniEvent. We train and deploy a multilingual T5<sub>BASE</sub> model for EE and a RoBERTa<sub>BASE</sub> model for event relation extraction. The website example is shown in Figure 3. The online system supports EE based on various English and Chinese classification schemata and ERE based on the MAVEN-ERE schema. The website mainly contains three parts. The input part includes a text entry field and several options. Users can choose the language, task, and ontology (i.e., classification schema) for event understanding. The results of EE are shown in the output field with extracted triggers and arguments

<sup>4</sup><https://omnievent.xlore.cn/>

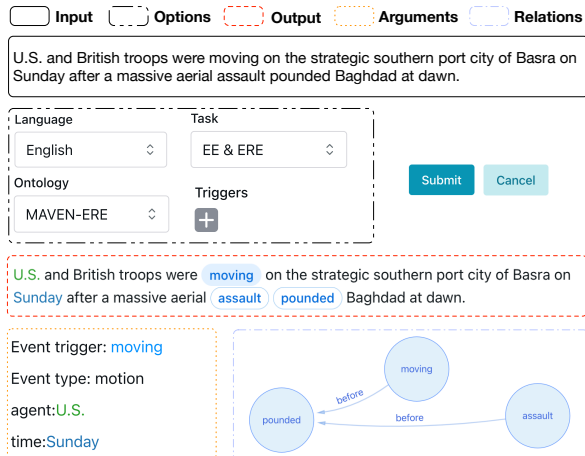


Figure 3: Example of the online demonstration. We re-arrange the layout of the website for a compact presentation. Better visualization in color.

| Task | Dataset       | CLS  | SL   | SP   | CG   |
|------|---------------|------|------|------|------|
| ED   | ACE 2005      | 68.6 | 68.6 | 71.0 | 66.0 |
|      | RichERE       | 51.4 | 50.1 | 50.4 | 51.4 |
|      | MAVEN         | 68.6 | 68.6 | 68.1 | 61.9 |
|      | ACE 2005 (ZH) | 75.8 | 75.9 | 73.5 | 71.6 |
|      | LEVEN         | 85.2 | 84.7 | 84.3 | 81.4 |
|      | FewFC         | 67.2 | 62.3 | 59.0 | 71.3 |
| EAE  | ACE 2005      | 58.7 | 49.4 | 40.1 | 45.7 |
|      | RichERE       | 68.3 | 59.7 | 24.3 | 24.9 |
|      | ACE 2005 (ZH) | 73.1 | 67.9 | 35.4 | 49.0 |
|      | FewFC         | 68.7 | 59.8 | 46.7 | 53.7 |

Table 3: Experimental results (F1,%) of implemented EE models in OmniEvent on various EE datasets. CLS: Classification; SL: Sequence labeling; SP: Span prediction; CG: Conditional generation. We evaluate the representative models: DMBERT, BERT+CRF, EEQA, and Text2Event for CLS, SL, SP, and CG, respectively.

highlighted. The results of ERE are shown as an event knowledge graph, where a node is an event and an edge is an identified relation between events. The example in Figure 3 shows the results of end-to-end event understanding (ED, EAE, and ERE) from the input plain text.

## 4 Evaluation

In this section, we conduct empirical experiments to evaluate the effectiveness of the OmniEvent toolkit on widely-used datasets.

### 4.1 Event Extraction

We evaluate the performance of representative EE models implemented in OmniEvent on various widely-used datasets. All the models are evaluated using the unified evaluation protocol, i.e., the

| Relation Type | Dataset        | P     | R    | F1   |
|---------------|----------------|-------|------|------|
| Coreference   | ACE 2005       | 94.5  | 81.7 | 87.7 |
|               | MAVEN-ERE      | 97.9  | 98.5 | 98.2 |
| Temporal      | TB-Dense       | 67.9  | 54.0 | 60.2 |
|               | MATRES         | 87.2  | 93.8 | 90.4 |
|               | TCR            | 78.3  | 78.3 | 78.3 |
|               | MAVEN-ERE      | 53.3  | 61.4 | 57.1 |
| Causal        | CausalTB       | 100.0 | 50.0 | 66.7 |
|               | EventStoryLine | 19.5  | 25.8 | 22.2 |
|               | MAVEN-ERE      | 36.0  | 26.4 | 30.5 |
| Subevent      | HiEve          | 21.4  | 13.4 | 16.5 |
|               | MAVEN-ERE      | 30.8  | 24.3 | 27.1 |

Table 4: Experimental results (%) of the implemented pairwise-based ERE model in OmniEvent on various ERE datasets. The backbone is RoBERTa<sub>BASE</sub>. The evaluation metric for coreference is B-cubed (Bagga and Baldwin, 1998).

output space is standardized and the results of EAE are from pipeline evaluation. The pre-processing script for ACE 2005 is the same as in Wadden et al. (2019). For EEQA, we utilize the same prompts as in the original paper for ACE 2005 and manually curate prompts for all the other datasets. The results of event detection and event argument extraction are shown in Table 3. The results demonstrate the effectiveness of OmniEvent, which achieves similar performance compared to their original implementations. OmniEvent provides all the experimental configuration files in the YAML format, which records all the hyper-parameters. Users can easily reproduce the results using the corresponding configuration files.

## 4.2 Event Relation Extraction

We also conduct empirical experiments to evaluate the performance of ERE models developed in OmniEvent on various widely-used datasets. As shown in Table 4, the results are on par or slightly better than the originally reported results in Wang et al. (2022), which demonstrates the validity of ERE models in OmniEvent. We also provide configuration files containing all the hyper-parameter settings for reproduction.

## 4.3 Experiments using LLMs

OmniEvent supports efficient fine-tuning and inference for LLMs. To examine the effectiveness and validity of LLMs support in OmniEvent and investigate the performance of models at different scales, we train a series of models on several datasets. Specifically, for ED and EAE, we fine-tune FLAN-

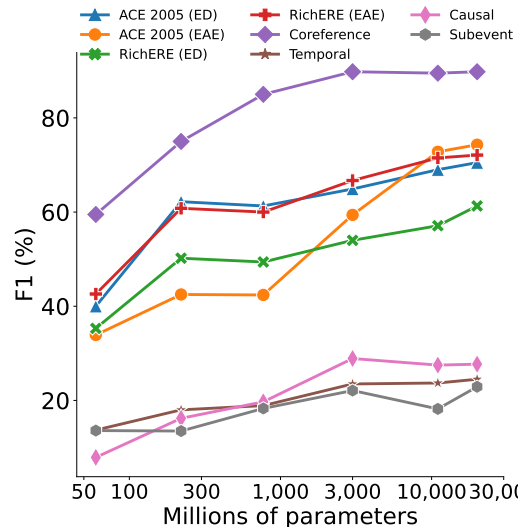


Figure 4: Experimental results of models at different scales on all event understanding tasks.

T5 (Wei et al., 2022) (from Small to XXL) and FLAN-UL2 (Tay et al., 2023), an LLM with 20 billion parameters on ACE 2005 and RichERE. For ERE, due to the lack of encoder-only LLMs, we use the same models as ED and EAE. We convert the ERE task into a sequence generation task. All the experiments are run on Nvidia A100 GPUs. Fine-tuning FLAN-UL2 on ACE 2005 consumes only about 25 GPU hours, which demonstrates the efficiency of LLMs support in OmniEvent. The results are shown in Figure 4. We can observe that larger models perform better and FLAN-UL2 achieves remarkable performance on ACE 2005 and RichERE datasets, which demonstrates the validity of LLMs support in OmniEvent. We can also notice that the results of ERE are much worse than the results in Table 4, which may be due to the extremely long contexts and complex output space of the ERE task. We hope the findings based on OmniEvent can inspire future research on how to better leverage LLMs for event understanding.

## 5 Conclusion and Future Work

In the paper, we present OmniEvent, a comprehensive, fair, and easy-to-use toolkit for event understanding. With the comprehensive and modular implementation, OmniEvent can help researchers and developers conveniently develop and deploy models. OmniEvent also releases several off-the-shelf models and deploys an online system for enhancing the applications of event understanding models. In the future, we will continually maintain OmniEvent to support more models and datasets.

## Limitations

The major limitations of OmniEvent are three-fold: (1) OmniEvent currently does not support document-level event extraction models and datasets, such as RAMS (Ebner et al., 2020) and WikiEvents (Li et al., 2021). OmniEvent also lacks support for a wider range of ERE models, such as constrained loss (Wang et al., 2020a) and ILP inference (Han et al., 2019). In the future, we will continue to maintain OmniEvent to support a broader range of models and datasets. (2) OmniEvent currently only supports two languages, Chinese and English, and does not yet support event relation extraction in Chinese. This might constrain the widespread usage of the OmniEvent toolkit. In the future, OmniEvent will support more languages. (3) Due to the limitations of training data and used models, the performance of our released model in practical applications is limited, especially in schemata and domains outside of the training data, such as the biomedical field. In the future, we will collect more training data and utilize advanced methods to develop more powerful and general models for event understanding.

## Ethical Considerations

We will discuss the ethical considerations and broader impact of this work here: (1) **Intellectual property.** OmniEvent is open-sourced and released under MIT license<sup>5</sup>. We adhere to the original licenses for all datasets and models used. Regarding the issue of data copyright, we do not provide the original data and we only provide processing scripts for the original data. (2) **Environmental Impact.** The experiments are conducted on the Nvidia A100 GPUs and consume approximately 350 GPU hours. This results in a substantial amount of carbon emissions, which incurs a negative influence on our environment (Strubell et al., 2019). (3) **Intended Use.** OmniEvent can be utilized to provide event understanding services for users, and it can also serve as a toolkit to assist researchers in developing and evaluating models. (4) **Misuse risks.** OmniEvent **should not** be utilized for processing and analyzing sensitive or uncopyrighted data. The output of OmniEvent is determined by the input text and **should not** be used to support financial or political claims.

<sup>5</sup><https://opensource.org/license/mit>

## Acknowledgements

This work is supported by a grant from the Institute for Guo Qiang, Tsinghua University (2019GQB0003) and the NSFC Youth Project (62006136). The authors thank all the anonymous reviewers for their detailed and valuable comments and suggestions. The authors also thank all the valuable issues on our GitHub repository.

## References

- Amit Bagga and Breck Baldwin. 1998. [Entity-based cross-document coreferencing using the vector space model](#). In *Proceedings of COLING-ACL*, pages 79–85. Morgan Kaufmann Publishers / ACL.
- Tommaso Caselli and Piek Vossen. 2017. [The event StoryLine corpus: A new benchmark for causal and temporal relation extraction](#). In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86.
- Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. [Dense event ordering with a multi-pass architecture](#). *Transactions of the Association for Computational Linguistics*, 2:273–284.
- Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. [Event extraction via dynamic multi-pooling convolutional neural networks](#). In *Proceedings of ACL-IJCNLP*, pages 167–176.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of NAACL-HLT*, pages 4171–4186.
- Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. [Deep learning for event-driven stock prediction](#). In *Proceedings of IJCAI*.
- Xinya Du and Claire Cardie. 2020. [Event extraction by answering \(almost\) natural questions](#). In *Proceedings of EMNLP*, pages 671–683.
- Xinya Du, Zixuan Zhang, Sha Li, Pengfei Yu, Hongwei Wang, Tuan Lai, Xudong Lin, Ziqi Wang, Iris Liu, Ben Zhou, et al. 2022. [Resin-11: Schema-guided event prediction for 11 newsworthy scenarios](#). In *Proceedings of NAACL-HLT: System Demonstrations*, pages 54–63.
- Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. [Multi-sentence argument linking](#). In *Proceedings of ACL*, pages 8057–8077.
- Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie M Strassel. 2015. [Overview of linguistic resources for the TAC KBP](#)

- 2015 evaluations: Methodologies and results. In *TAC*.
- Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie M Strassel. 2016. [Overview of Linguistic Resources for the TAC KBP 2016 Evaluations: Methodologies and Results](#). In *TAC*.
- Joe Ellis, Jeremy Getman, and Stephanie M Strassel. 2014. [Overview of linguistic resources for the TAC KBP 2014 evaluations: Planning, execution, and results](#). In *TAC*.
- Jeremy Getman, Joe Ellis, Zhiyi Song, Jennifer Tracey, and Stephanie Strassel. 2017. [Overview of linguistic resources for the tac kbp 2017 evaluations: Methodologies and results](#). In *TAC*.
- Goran Glavaš, Jan Šnajder, Marie-Francine Moens, and Parisa Kordjamshidi. 2014. [HiEve: A corpus for extracting event hierarchies from news stories](#). In *Proceedings of LREC*, pages 3678–3683.
- Rujun Han, Qiang Ning, and Nanyun Peng. 2019. [Joint event and temporal relation extraction with shared representations and structured prediction](#). In *Proceedings of EMNLP-IJCNLP*, pages 434–444.
- Quzhe Huang, Yutong Hu, Shengqi Zhu, Yansong Feng, Chang Liu, and Dongyan Zhao. 2023. [More than classification: A unified framework for event temporal relation extraction](#). In *Proceedings of ACL*.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. [Imagenet classification with deep convolutional neural networks](#). In *Proceedings of NeurIPS*, pages 1106–1114.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. [Conditional random fields: Probabilistic models for segmenting and labeling sequence data](#). In *Proceedings of ICML*, pages 282–289.
- Sha Li, Heng Ji, and Jiawei Han. 2021. [Document-level event argument extraction by conditional generation](#). In *Proceedings of NAACL-HLT*, pages 894–908.
- Xinyu Li, Fayuan Li, Lu Pan, Yuguang Chen, Weihua Peng, Quan Wang, Yajuan Lyu, and Yong Zhu. 2020. [DuEE: A large-scale dataset for chinese event extraction in real-world scenarios](#). In *Proceedings of NLPCC*, pages 534–545.
- Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. [A joint neural model for information extraction with global features](#). In *Proceedings of ACL*, pages 7999–8009.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. [Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction](#). In *Proceedings of ACL-IJCNLP*, pages 2795–2806.
- Mingyu Derek Ma, Jiao Sun, Mu Yang, Kung-Hsiang Huang, Nuan Wen, Shikhar Singh, Rujun Han, and Nanyun Peng. 2021. [Eventplus: A temporal event understanding pipeline](#). In *Proceedings of NAACL-HLT*, pages 56–65.
- Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. [Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction](#). In *Proceedings of ACL*.
- Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. [Annotating causality in the TempEval-3 corpus](#). In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19.
- Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. [Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks](#). In *Proceedings of NAACL-HLT*, pages 27–38.
- Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. [Joint reasoning for temporal and causal relations](#). In *Proceedings of ACL*, pages 2278–2288.
- Qiang Ning, Hao Wu, and Dan Roth. 2018b. [A multi-axis annotation scheme for event temporal relations](#). In *Proceedings of ACL*, pages 1318–1328.
- Hao Peng, Xiaozhi Wang, Feng Yao, Kaisheng Zeng, Lei Hou, Juanzi Li, Zhiyuan Liu, and Weixing Shen. 2023. [The devil is in the details: On the pitfalls of event extraction evaluation](#). In *Findings of ACL*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. [Language models are unsupervised multitask learners](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *J. Mach. Learn. Res.*, 21:140:1–140:67.
- 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 ACM SIGKDD, KDD '20*, page 3505–3506.
- Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. [From light to rich ere: annotation of entities, relations, and events](#). In *Proceedings of the the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. [Energy and policy considerations for deep learning in NLP](#). In *Proceedings of ACL*, pages 3645–3650.
- Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. [UL2: Unifying language learning paradigms](#). In *Proceedings of ICLR*.
- Piek Vossen, Rodrigo Agerri, Itziar Aldabe, Agata Cybulska, Marieke van Erp, Antske Fokkens, Egoitz Laparra, Anne-Lyse Minard, Alessio Palmero Aprosio, German Rigau, Marco Rospocher, and Roxane Segers. 2016. [Newsreader: Using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news](#). *Knowl. Based Syst.*, 110:60–85.
- David Wadden, Ulme Wennberg, Yi Luan, and Hananeh Hajishirzi. 2019. [Entity, relation, and event extraction with contextualized span representations](#). In *Proceedings of EMNLP-IJCNLP*, pages 5784–5789.
- Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. [ACE 2005 multilingual training corpus](#). *Linguistic Data Consortium*, 57.
- Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020a. [Joint constrained learning for event-event relation extraction](#). In *Proceedings of EMNLP*, pages 696–706.
- Shichao Wang, Xiangrui Cai, Hongbin Wang, and Xiaojie Yuan. 2021a. [Incorporating circumstances into narrative event prediction](#). In *Findings of EMNLP*, pages 4840–4849.
- Xiaozhi Wang, Yulin Chen, Ning Ding, Hao Peng, Zimu Wang, Yankai Lin, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, et al. 2022. [MAVEN-ERE: A unified large-scale dataset for event coreference, temporal, causal, and subevent relation extraction](#). In *Proceedings of EMNLP*, pages 926–941.
- Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. [Adversarial Training for Weakly Supervised Event Detection](#). In *Proceedings of NAACL-HLT*, pages 998–1008.
- Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020b. [MAVEN: A Massive General Domain Event Detection Dataset](#). In *Proceedings of EMNLP*, pages 1652–1671.
- Ziqi Wang, Xiaozhi Wang, Xu Han, Yankai Lin, Lei Hou, Zhiyuan Liu, Peng Li, Juanzi Li, and Jie Zhou. 2021b. [CLEVE: Contrastive Pre-training for Event Extraction](#). In *Proceedings of ACL-IJCNLP*, pages 6283–6297.
- Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. [Finetuned language models are zero-shot learners](#). In *Proceedings of ICLR*.
- 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 EMNLP: System Demonstrations*, pages 38–45.
- Susmitha Wunnava, Xiao Qin, Tabassum Kakar, Cansu Sen, Elke A Rundensteiner, and Xiangnan Kong. 2019. [Adverse drug event detection from electronic health records using hierarchical recurrent neural networks with dual-level embedding](#). *Drug safety*, 42:113–122.
- 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 NAACL-HLT*, pages 483–498.
- Feng Yao, Chaojun Xiao, Xiaozhi Wang, Zhiyuan Liu, Lei Hou, Cunchao Tu, Juanzi Li, Yun Liu, Weixing Shen, and Maosong Sun. 2022. [LEVEN: A large-scale chinese legal event detection dataset](#). In *Findings of ACL*, pages 183–201.
- Zhenrui Yue, Huimin Zeng, Mengfei Lan, Heng Ji, and Dong Wang. 2023. [Zero-and few-shot event detection via prompt-based meta learning](#). In *Proceedings of ACL*.
- Ningyu Zhang, Shumin Deng, Zhen Bi, Haiyang Yu, Jiacheng Yang, Mosha Chen, Fei Huang, Wei Zhang, and Huajun Chen. 2020. [Openue: An open toolkit of universal extraction from text](#). In *Proceedings of EMNLP: system demonstrations*, pages 1–8.
- Ningyu Zhang, Xin Xu, Liankuan Tao, Haiyang Yu, Hongbin Ye, Shuofei Qiao, Xin Xie, Xiang Chen, Zhoubo Li, and Lei Li. 2022. [DeepKE: A deep learning based knowledge extraction toolkit for knowledge base population](#). In *Proceedings of EMNLP: System Demonstrations*, pages 98–108.
- Yang Zhou, Yubo Chen, Jun Zhao, Yin Wu, Jiexin Xu, and Jinlong Li. 2021. [What the role is vs. what plays the role: Semi-supervised event argument extraction via dual question answering](#). In *Proceedings of AAAI*.



## Appendices

### A Unified Data Format

An instance converted to the unified data format is shown in Code 3. The data format comprehensively records all the event-related information: triggers, arguments, and coreference, temporal, causal, and subevent relations.

```
{ # one instance
  "id": "instance.001.01",
  "text": "U.S. and British troops were moving on the strategic southern port city of Basra Saturday on Sunday after a massive aerial assault pounded Baghdad at dawn .",
  "events": [
    {
      "type": "attack",
      "triggers": { # triggers that have a coreference relation with each other
        "id": "trigger1",
        "trigger_word": "assault",
        "offset": [22, 23],
        "arguments": [
          {"mention": "U.S.", "offset": [0, 1], "role": "attacker"},
          {"mention": "British", "offset": [1, 2], "role": "attacker"},
          {"mention": "dawn", "offset": [26, 27], "role": "time"}
        ]
      }
    },
    {
      "type": "motion",
      "triggers": {
        "id": "trigger2",
        "trigger_word": "moving",
        "offset": [5, 6],
        "arguments": [
          {"mention": "Sunday", "offset": [17, 18], "role": "time"},
        ]
      }
    }
  ],
  # .....
  "event-relations": {
    "temporal": [
      ["trigger1", "before", "trigger2"]
    ],
    "causal": [],
    "subevent": []
  }
}
```

Code 3: An instance with the unified data format. The triggers that recorded in an item of “events” have a coreference relation with each other.

# CocoSciSum: A Scientific Summarization Toolkit with Compositional Controllability

Yixi Ding<sup>†</sup> Yanxia Qin<sup>†\*</sup> Qian Liu<sup>‡</sup> Min-Yen Kan<sup>†</sup>

<sup>†</sup>School of Computing, National University of Singapore

<sup>‡</sup>Sea AI Lab

{yixi.d, qinyx, kanmy}@comp.nus.edu.sg, liuqian@sea.com

## Abstract

We present a novel toolkit for controlled summarization of scientific documents, designed for the specific needs of the scientific community. Our system generates summaries based on user preferences, adjusting key attributes specifically of length and keyword inclusion. A distinguishing feature is its ability to manage multiple attributes concurrently, demonstrating **Compositional Controllability** for **Scientific Summarization** (*CocoSciSum*). Benchmarked against the strong Flan-T5 baseline, *CocoSciSum* exhibits superior performance on both the quality of summaries generated and the control over single and multiple attributes. Moreover, *CocoSciSum* is a user-centric toolkit, supporting user preferences expressed in natural language instructions, and accommodating diverse input document formats. *CocoSciSum* is available on GitHub<sup>1</sup> with an introduction video<sup>2</sup>.

## 1 Introduction

Scientific summarization refers to the process of distilling scientific documents such as research papers into shorter versions that capture the key information. It has become increasingly important as it can facilitate quick search results filtering, as seen in Semantic Scholar’s TL;DR (Too Long, Didn’t Read) feature. However, we argue that controlled summarization for scientific documents, which generates user-customized summaries over various control attributes, can further enhance personalized result filtering and paper comprehension. For example, a reader primarily interested in a paper’s relationship to a particular term — say, ‘*cloze task*’ — can instruct the summarization system to focus on the term when generating the summary. Accordingly, we introduce the first toolkit designed for

controlled summarization in the scientific domain. It consistently generates high-quality summaries, as validated by various automatic evaluation metrics and human annotators. Aside from improved performance, our system, *CocoSciSum*, makes two key contributions to the summarization landscape.

**Contribution 1: Compositional Controllability.** We seek a unified solution for all summarization functions, aiming to reduce memory usage and streamline deployment. *CocoSciSum* provides three modes of summarization: (1) vanilla summarization, (2) single-attribute controlled summarization (managing either length or keyword inclusion), and (3) compositionally-controlled summarization, which jointly regulates both length and keywords. It employs instruction-tuned PLMs; specifically adopting FLAN-T5 (Chung et al., 2022) as the backbone model. Figure 1 shows the framework of the summarization model.

To achieve such compositional goals, past summarization systems all have resorted to separate models to control each attribute of interest (Takase and Okazaki, 2019; Saito et al., 2020; Liu et al., 2022; Zheng et al., 2020; Narayan et al., 2021). However, this presents a significant challenge to the pre-trained language model (PLM) finetuning paradigm, due to the scarcity of training data. The necessary creation of training data incurs significant annotation costs, given the complexity of annotating multiple reference summaries for a single document, each conforming to individual or multiple attribute constraints. The disparity between available and desired training data compels us to seek an alternative, as reference summaries for multiple controls are not available and prohibitively expensive to annotate. Instead, we break down the annotation task into simpler sub-tasks, synthesizing summaries constrained for each attribute separately. We hypothesize that by initially training the model on simplified relevant tasks, it can subsequently generalize. That is, we finetune the PLM using

\*Corresponding author

<sup>1</sup><https://github.com/WING-NUS/SciAssist/tree/CocoSciSum>

<sup>2</sup><https://youtu.be/YC1YDeEjAbQ>

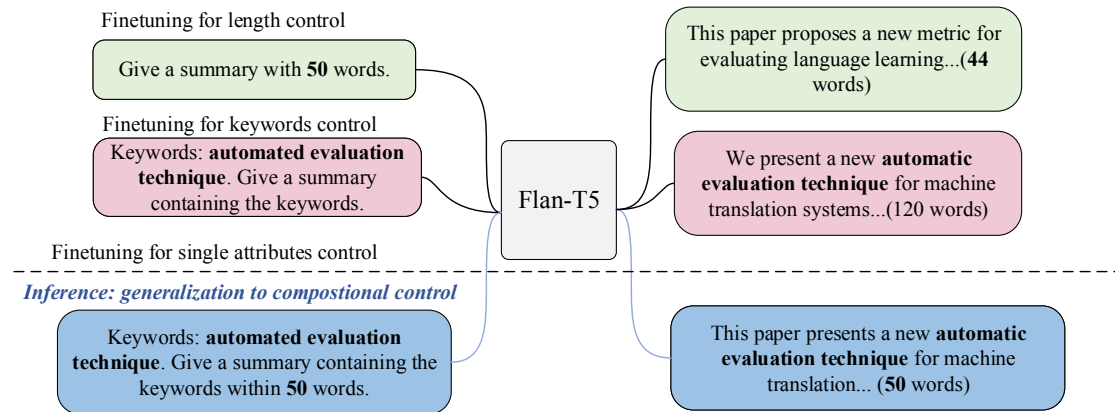


Figure 1: The summarization model with compositional controllability used in *CocoSciSum*. The FLAN-T5 backbone model is separately finetuned for length and keyword controls, and generalizes to the composition task. Control attribute mentions are highlighted in bold.

single-attribute constrained data, expecting its capability to solve composite summarization tasks.

**Contribution 2: User-friendly.** We build *CocoSciSum* to allow easy uptake for those simply curious as well as serious practitioners. We offer a demonstration page<sup>3</sup> for *CocoSciSum*’s immediate utilization, and an easy-to-install Python package along with comprehensive development documentation<sup>4</sup> for practitioners.

*CocoSciSum* itself is also user-friendly: it interprets user preferences specified in natural language instructions and supports multiple input document formats. In contrast to previous work that employed opaque vectors (i.e., unreadable to humans) for summarization control, *CocoSciSum* adopts user-friendly natural language to specify instructions, such as “Give a summary within 50 words and containing the phrase ‘automated evaluation technique’.” (Figure 1). Finally, considering the prevalence of portable document format (PDF) for scientific documents (which are not directly machine-readable), we incorporate a specialized PDF text extraction module. Our toolkit also supports other document formats like plain text and structured JSON for wider practical usage.

## 2 Architecture

We show the architecture of *CocoSciSum* in Figure 2. It consists of training and inference phases.

<sup>3</sup><https://huggingface.co/spaces/wing-nus/SciAssist>

<sup>4</sup><https://wing-sciassist.readthedocs.io/en/latest/>

## 2.1 Training

**Implementation Framework.** PyTorch Lightning<sup>5</sup> is a flexible high-level interface for PyTorch, abstracting complex training logic for rapid prototyping. We adopt Lightning for *CocoSciSum*’s workflow. The workflow of *CocoSciSum* starts with a `LightningDataModule` to download and split datasets and then assemble batched vector data. Subsequently, a `LightningModule` that encapsulates the model and optimizer is used. Simultaneously, a `Trainer` coordinates the above components to automate training.

**Configuration.** Hydra (Yadan, 2019) is an open-source Python framework that simplifies the management of configurations in applications. In particular, Hydra allows dynamically creating a hierarchical configuration by composition from multiple sources, and overriding it through both the original configuration files and the command line, facilitating specifying hyper-parameters and experimental settings. We follow a deep learning project template<sup>6</sup> for designing Hydra in *CocoSciSum*.

**Utility.** The `Utility` module centralizes all data processing procedures, inclusive of tokenizing text-label mixed strings (`tokenize_and_align_labels`), generating the text sequence from a matrix of probabilities over a vocabulary (`prob2text`) and customizing batch collation (`batch_collator`). All data utility functions are encapsulated into a `DataUtils` class,

<sup>5</sup><https://lightning.ai/pytorch-lightning>

<sup>6</sup><https://github.com/ashleve/lightning-hydra-template>

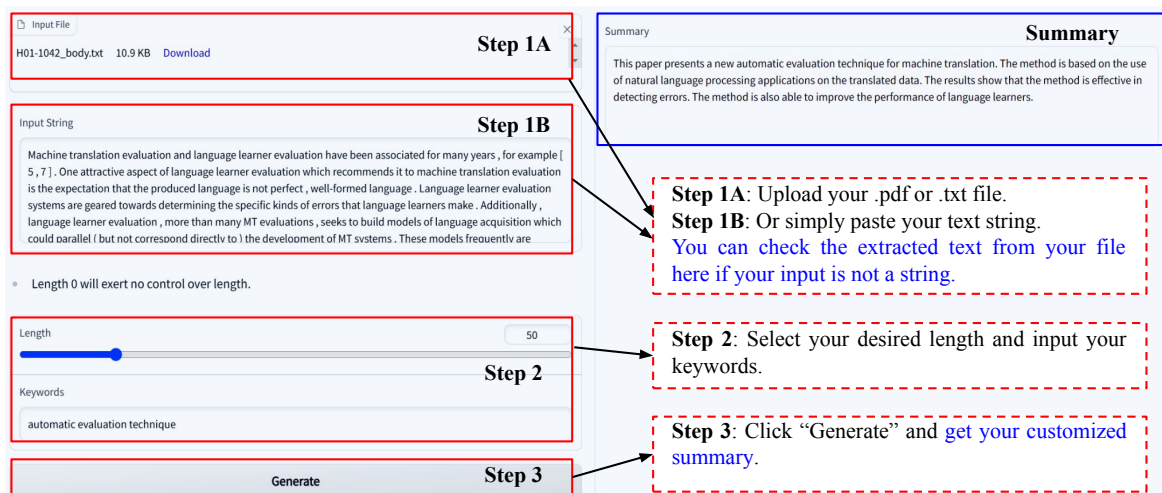


Figure 2: *CocoSciSum*'s interface and usage for controlled summarization. It accepts multiple formats for input documents, including PDF, text string, and text files. User preferences on length, keyword inclusion or both can be specified in the control section. The customized summary will be presented in the output summary zone with a click on the Generate button.

which facilitates the use of different datasets in various formats, and enables the seamless application of a single dataset to multiple models.

**Adding New Datasets.** Integrating additional datasets for use in *CocoSciSum* is simple. We create a new `DataModule` class inheriting from `LightningDataModule`, implementing several key functions to download the data (`prepare_data`), split it into training and validation sets (`setup`), prepare batched vector data (`train_dataloader`). A Hydra file associated with the dataset can be used for easy configuration.

**Adding New Models.** Users can also add a new model by implementing a model class inherited from `LightningModule`. Such a class can customize the training logic, defined in `training_step` function. A Hydra configuration file can be used to define model settings and hyperparameters.

## 2.2 Inference

In addition to supporting various input document formats *CocoSciSum* is designed to be easily deployable on Linux, Windows, and MacOS, further enhancing its usability. Due to the absence of a universal toolkit for all three OS, we use different text extraction toolkits for PDF files on different platforms. Moreover, users can access *CocoSciSum* through its Python package or a demo page 2.3. Accordingly, the system offers both direct summaries and text-summary pairs for different scenarios. On

the demo page, we fix the control prompts to be the instructions used for finetuning the model in Section 3, for generating summaries in high quality. Meanwhile, the provided model in the Python package, sourced from Section 3, supports more flexible natural language prompts.

## 2.3 Demonstration

To illustrate the functionality of *CocoSciSum*, we offer a demo page featuring a user-friendly interface, as shown in Figure 2. The interface is divided into two sections. The left side serves as the user input area, which accepts input documents, user-specified control attributes, and the generation action. The right side shows the system-generated summary.

Users can customize summary length using a slider with increments of 50 or input specific keywords if necessary. The system also displays the text extracted from the input document in a text box, offering users a chance to verify the content, thereby enhancing overall usability.

Our demo page is hosted on the Huggingface platform, which provides various hardware deployment options. We have utilized the free CPU-basic version for our demo, equipped with 2 CPUs and a total of 16 GB RAM. On average, a summary is generated in less than 30 seconds, indicating that *CocoSciSum* can be deployed effortlessly, even on hardware with limited resources. This is mainly due to the small-sized model, which allows for satisfactory performance with a reasonable time cost.

### 3 Compositionally Controlled Summarization

We now introduce the compositionally-controlled summarization model used in *CocoSciSum*. The model is designed to support all summarization functions, including standard, single-attribute-controlled and multi-attribute-controlled summarization. We first introduce the backbone summarization model and then detail the instruction tuning for compositional controllability.

#### 3.1 Backbone Model

Previous controlled summarization work (He et al., 2022; Zhang et al., 2022) has successfully leveraged the PLM finetuning paradigm, which mostly utilizes text generation PLMs such as BART (Lewis et al., 2019) and T5 (Raffel et al., 2020). However, the lack of labeled data and high annotation costs for the compositionally-controlled summarization motivates us to seek another solution. Drawing inspiration from instruction tuning (Wei et al., 2021; Chung et al., 2022), we utilize an instruction-tuned PLM to harness its zero-shot performance on unseen tasks. Specifically, we instruction-tune a PLM on two single-attribute-controlled summarization tasks, anticipating its zero-shot capability on multi-attribute-controlled summarization.

We employ the FLAN-T5 model, which is based on the T5 model and instruction tuned on 1,836 tasks. As described in the prior work (Chung et al., 2022), this wide-ranging finetuning results in substantially improved zero-shot performance on unseen tasks. In addition, among these instruction tuning tasks, 1,554 of them are natural instruction tasks, further enhancing its comprehension of natural language instructions. We capitalize on this and use natural language to prompt FLAN-T5 for user-desired summaries, such as “Summarize the text with 50 words:”.

#### 3.2 Finetuning

To further finetuning FLAN-T5 on single-attribute controlled summarization tasks, we synthesize their finetuning data from a generic summarization dataset in the scientific domain.

**Length Control.** Compared to coarse-grained length control in previous work such as “long” and “short”, we aim to generate summaries with exact length control. We propose to use the following prompt: “Give a summary of the following

text, which has less than  $n$  words:” to control the length of the system-generated summary, where  $n$  is the number of words in the reference summary. To avoid the sparseness of exact  $n$  in finetuning, we round it up to the nearest bin of 50, e.g.,  $167 \rightarrow 200$ .

**Keyword Control.** The purpose of keyword control is to generate summaries relevant to the keywords of interest. We use the following instruction to guide the model: “Keywords:  $[k_1, k_2, \dots]$ . Give a summary of the following text based on these keywords: ”, where  $k$  refers to a keyword. Here, a keyword can be a single word or a phrase, used to represent a scientific term. Given a generic summarization data instance including a document–summary pair, we lack the keywords that appeared in the summary. Thus we propose to extract keywords from the reference summary. Again, we take advantage of FLAN-T5 for its zero-shot performance in information extraction tasks and prompt FLAN-T5 to generate keywords in the reference summaries. In particular, we instruct a FLAN-T5-XL<sup>7</sup> model with the following prompt “What keywords does this scientific summary include? [Ref Summary] Keywords:”.

**Compositional Control.** Given the limitations in available computing resources, we adopt a small-sized PLM, FLAN-T5-base with 230 million parameters, to finetune with length-controlled and keyword-controlled data expecting compositional controllability. Due to its limited model capacity, the FLAN-T5-base model is unlikely to keep its first gained ability when finetuned for the second ability (Fu et al., 2023). We validated this phenomenon also occurs in our experiments, which guides us for a different finetuning strategy for compositional control. Inspired by (Cachola et al., 2020), we shuffle two finetuning datasets and then train the model with the shuffled dataset to generalize on both attributes.

## 4 Experiments

### 4.1 Experimental Settings

**Datasets.** We construct datasets using two scientific domain summarization datasets: *MuP* (Cohan et al., 2022) and *SciSumm* (Chandrasekaran et al.,

<sup>7</sup>The FLAN-T5-XL model is used due to better performance.

| Model                  | R-1          | R-2         | R-L          | R-LSum       | BScore       | MAD↓        | PCC↑        | SR↑         | FACT↑       |
|------------------------|--------------|-------------|--------------|--------------|--------------|-------------|-------------|-------------|-------------|
| T5                     | 31.10        | 5.67        | 17.98        | 27.99        | 0.091        | –           | –           | –           | –           |
| FLAN-T5                | 33.59        | 6.40        | 19.34        | 29.60        | 0.128        | –           | –           | –           | –           |
| Ours                   | 33.97        | 6.54        | <i>19.48</i> | 30.00        | <i>0.131</i> | –           | –           | –           | –           |
| FLAN-T5 <sub>Len</sub> | 33.33        | 6.49        | 19.43        | 29.44        | 0.129        | 1.41        | -0.01       | –           | –           |
| Ours <sub>Len</sub>    | <i>35.24</i> | 6.80        | <b>19.92</b> | <i>31.12</i> | <b>0.133</b> | <b>0.36</b> | <b>0.85</b> | –           | –           |
| CTRLsum <sub>Kw</sub>  | 32.00        | <b>7.93</b> | 18.06        | 28.62        | 0.090        | –           | –           | <b>0.61</b> | –           |
| FLAN-T5 <sub>Kw</sub>  | 33.08        | 6.53        | 19.41        | 29.23        | 0.130        | –           | –           | 0.24        | 0.44        |
| Ours <sub>Kw</sub>     | 34.43        | 7.06        | 18.97        | 30.34        | 0.129        | –           | –           | 0.57        | 0.58        |
| T5 <sub>Len,Kw</sub>   | 31.20        | 5.56        | 18.07        | 28.11        | 0.085        | 1.46        | 0.00        | 0.23        | –           |
| Ours <sub>Len,Kw</sub> | <b>35.35</b> | <i>7.16</i> | 19.45        | <b>31.26</b> | 0.130        | <i>0.60</i> | <i>0.83</i> | <i>0.58</i> | <b>0.65</b> |

Table 1: Experimental results. R-\* (ROUGE), BScore (BERTScore), MAD and PCC are calculated on *MuP-dev-1k*, SR and FACT are calculated on *KW-test*. The settings denoted with *Len* and *Kw* are attribute-controlled inferences, others are vanilla summarization inferences. Optimal results are highlighted in **bold**. *Italicization* indicates values that are the second best.

2019). A training dataset *Coco-train* and a development dataset *Coco-dev* are created for model finetuning (See Section 3.2). Two test sets *MuP-dev-1k* and *KW-test* are constructed for evaluating summarization quality and length controllability, and keyword controllability, respectively. Details and statistics are presented in Appendix A.

**Evaluation Metrics.** We evaluate the output summaries for both summarization quality and attribute controllability. We employ ROUGE (Lin, 2004) and BERTScore (Zhang et al., 2020) for quality evaluation. For length controllability evaluation, we adopt the Mean of Absolute Deviation (MAD, Liu et al., 2018) and the Pearson Correlation Coefficient (PCC, Liu et al., 2018) between length codes. For evaluating keyword controllability, we use the Success Rate (SR, Fan et al., 2018; He et al., 2022) and the Factual Correctness (FACT, Krishna et al., 2023). More details are in Appendix B.

**Baselines.** We choose baseline methods including a Seq2Seq model T5 (Raffel et al., 2020), a controlled summarization method CTRLsum (He et al., 2022), and an instruction-tuned model FLAN-T5. The FLAN-T5 baseline is finetuned with the general summarization dataset *MuP*. The T5 models is finetuned with the same attribute-controlled dataset *Coco-train* as our model, CTRLsum is finetuned with keyword-controlled dataset *KW-data*. We have chosen not to include scientific PLMs, such as SciBERT (Beltagy et al., 2019) and Galactica (Taylor et al., 2022) because a fair comparison is currently not feasible (wrong model size, architecture, and inability for instruction tuning).

## 4.2 Results

We present the experimental results in Table 1.

**Summary Quality.** Compared to T5 in the vanilla summarization setting, both FLAN-T5 and our model demonstrate improved performances across all quality evaluation metrics (e.g., 33.59 and 33.97 V.S. 31.1 on R-1). This indicates the superiority of FLAN-T5 over T5. However, our system consistently surpasses the strong FLAN-T5 baseline in both standard and controlled settings, with an average performance increase of 1.21 on R-1, underlining its effectiveness. Furthermore, by comparing our model in three controlled inferences with the vanilla setting, we observe rises in ROUGE scores (an average increase of 1.04 on R-1). This implies that control signals further enhance the quality of the generated summaries. Interestingly, this advantage is exclusive to our model and not observed in either T5 or FLAN-T5.

**Length Controllability.** From Table 1, the low PCC value (-0.01) of the FLAN-T5 baseline indicates that it has no controllability over the length of summaries, suggesting it does not understand unseen instructions of length control. We observe that T5 has no controllability over length (PCC 0), even trained with length-controlled data. While our model in two length-controlled inferences (*Len* and *Len,Kw*) achieve high PCC values ( $\geq 0.83$ , out of 1), showing the presence of strong length controllability in *CocoSciSum*. Interestingly, we observe weaker controllability of our model in the compositional control setting *Len,Kw*, compared with single length control. We assume that multiple control

| System                 | Attribute                     | Summary (# of Words)  |
|------------------------|-------------------------------|---|
| FLAN-T5 <sub>Kw</sub>  | “Cloze task” ✗                | This paper proposes a new method for pre-training language models. The method is based on a masked language model (LM) pre-training objective. The masked language models are used to predict the original vocabulary id of the masked word. The authors show that the proposed method can achieve state-of-the-art performance on a range of tasks. (54)                           |
| Ours                   | —                             | This paper proposes a bidirectional language model pre-training method for NLP tasks. The proposed method is based on a masked language model (MLM) pre-training objective. The masked language models randomly mask some of the tokens from the input ... The proposed method achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks. (90) |
| Ours <sub>Len</sub>    | 50 words ✓                    | This paper proposes a bidirectional language model pretraining method. The proposed method is based on a masked language model. The proposed method achieves state-of-the-art performance on several NLP tasks. (29)  |
| Ours <sub>Kw</sub>     | “Cloze task” ✓                | We present a new pre-training method for language representations. The method is inspired by the <b>Cloze task</b> , which is a task-specific task that requires pre-trained language models to perform well ... model is able to achieve state-of-the-art performance on a large set of sentence-level tasks, and outperforms many task-specific baselines. (125)                  |
| Ours <sub>Len,Kw</sub> | 50 words ✓;<br>“Cloze task” ✓ | This paper proposes a bidirectional pre-training method for language representations. The method is inspired by the <b>Cloze task</b> . The method is evaluated on a large suite of sentence-level and token-level tasks. (31)  |

Table 2: Example summaries of BERT, generated by different systems. Attribute denotes the control attribute mentions, “50” means to limit the summary within 50 words, and “Cloze task” is the user-preferred keyword. Evidences of controllability are highlighted in bold. ✓ and ✗ indicate successful and failed control, respectively.

signals are difficult to understand for a relatively small-sized PLM with 230 million parameters compared to those with 7 billion or more parameters.

**Keyword Controllability.** With the success rate, we evaluate whether the model can generate a summary containing user-predefined keywords following the natural language instruction. A success rate of 0.24 for FLAN-T5 indicates it has only limited controllability for keywords. In contrast, our model achieves a higher success rate of 0.57, offering more than double the probability of generating summaries inclusive of keywords, thus demonstrating our model’s superior and more reliable keyword controllability. CTRLsum achieves the best controllability over keywords among all methods, however no length controllability as it is controlled by the number of keywords.

Our approach not only aims to include keywords in the summary but also strives to prevent overfitting to the success rate by generating fictional summaries. Compared to the factual correctness of 0.44 of FLAN-T5, our model yields a higher score of 0.58, further reinforcing its superior controllability. When comparing our system in the compositionally-controlled setting  $Len,Kw$  with T5, which show no or random controllability (0 for PCC, 0.23 for SR), it is apparent that FLAN-T5 serves as the foundational model for the compositional controllability of our model.

**Case Study.** We present several system-generated summaries for the BERT (Devlin et al.,

2019) paper in Table 2. All summaries are fluent, grammar error-free, and easy to understand, which shows strong summarization abilities of Flan-T5 models again. More importantly, we observe controllability over both length and keywords from our model through the desired length and keyword inclusion. Another interesting observation is that all summaries include the main contribution (“*a new pre-training method for language representations*”) as well as the experimental information (“state-of-the-art performance”) of the source scientific document.

## 5 Conclusion

We present the first scientific toolkit, *CocoSciSum*, which summarizes scientific papers by customizing multiple aspects, including fine-grained length control, keyword inclusion, and compositional control of both. The user preferences are specified in natural language instructions rather than human-unreadable vectors. We provide an easy-to-install Python package and a demo page for different uses. *CocoSciSum* supports multiple input formats of scientific papers, plain text, structured JSON, and the most commonly used PDF. We employ FLAN-T5 as the backbone model, finetuning it on single-attribute-controlled tasks, and prove its capability for multi-attribute control.

In future work, we will explore compositional controllability conflicts and user-specified weighting between multiple attributes.

## Acknowledgements

We thank the reviewers for their valuable feedback and suggestions. This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (T1 251RES2216, T1 251RES2037).

## 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.
- Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel Weld. 2020. **TLDR: Extreme summarization of scientific documents**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4766–4777, Online. Association for Computational Linguistics.
- Muthu Kumar Chandrasekaran, Michihiro Yasunaga, Dragomir Radev, Dayne Freitag, and Min-Yen Kan. 2019. **Overview and results: CI-scisumm shared task 2019**.
- 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**.
- Arman Cohan, Guy Feigenblat, Tirthankar Ghosal, and Michal Shmueli-Scheuer. 2022. **Overview of the first shared task on multi perspective scientific document summarization (MuP)**. In *Proceedings of the Third Workshop on Scholarly Document Processing*, pages 263–267, Gyeongju, Republic of Korea. 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, David Grangier, and Michael Auli. 2018. **Controllable abstractive summarization**. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.
- Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. **Specializing smaller language models towards multi-step reasoning**.
- Junxian He, Wojciech Kryscinski, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2022. **CTRL-sum: Towards generic controllable text summarization**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5879–5915, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. **Longeval: Guidelines for human evaluation of faithfulness in long-form summarization**.
- 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**.
- 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.
- Yizhu Liu, Qi Jia, and Kenny Zhu. 2022. **Length control in abstractive summarization by pretraining information selection**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6885–6895, Dublin, Ireland. Association for Computational Linguistics.
- Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. **Controlling length in abstractive summarization using a convolutional neural network**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119, Brussels, Belgium. Association for Computational Linguistics.
- Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. **Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction**. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.
- Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan McDonald. 2021. **Planning with learned entity prompts for abstractive summarization**. *Transactions of the Association for Computational Linguistics*, 9:1475–1492.
- 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**.



Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, Atsushi Otsuka, Hisako Asano, Junji Tomita, Hiroyuki Shindo, and Yuji Matsumoto. 2020. [Length-controllable abstractive summarization by guiding with summary prototype](#). *CoRR*, abs/2001.07331.

Sho Takase and Naoaki Okazaki. 2019. [Positional encoding to control output sequence length](#). 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 3999–4004, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. [Finetuned language models are zero-shot learners](#). *CoRR*, abs/2109.01652.

Omry Yadan. 2019. [Hydra - a framework for elegantly configuring complex applications](#). Github.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#).

Yusen Zhang, Yang Liu, Ziyi Yang, Yuwei Fang, Yulong Chen, Dragomir Radev, Chenguang Zhu, Michael Zeng, and Rui Zhang. 2022. [Macsum: Controllable summarization with mixed attributes](#).

Changmeng Zheng, Yi Cai, Guanjie Zhang, and Qing Li. 2020. [Controllable abstractive sentence summarization with guiding entities](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5668–5678, Barcelona, Spain (Online). International Committee on Computational Linguistics.

## A Dataset Construction and statistics

In this section, we introduce details about the dataset construction.

**Training Data** Our training data is derived from two generic summarization datasets in the scientific domain: *MuP* (Cohan et al., 2022) and *SciSumm* (Chandrasekaran et al., 2019). We adopt these two datasets as they both provide reference summaries and are large-scale (18.9k and 1k document-summary pairs, respectively). We construct length-controlled labeled data from the training set of *MuP-train* with the method outlined in Section 3.2, denoted as *LEN-data*. The keyword-controlled data comes from *SciSumm*, denoted as *KW-data*. Finally,

*LEN-data* and *KW-data* are merged together and then randomly split into a training set *Coco-train* and a development set *Coco-dev* using a 9:1 ratio.

**Test sets** We design three test sets for different testing objectives. For calculating ROUGE scores and BERTScore, length controllability metrics MAD and PCC, we randomly sampled 1000 data points from *MuP*’s development set, creating our test set (*MuP-dev-1k*).

To ensure high-quality keywords for calculating keyword controllability evaluation metric Success Rate, we choose SciERC (Luan et al., 2018), a dataset with human-annotated scientific entities, as the data source. From SciERC, we select entities classified as "Method" and "Task", each comprising less than 3 words, to simulate user preferences. This results in a set of 229 (document, keyword) pairs from 55 documents, denoted as *KW-test*.

To facilitate the manual evaluation of Factual Correctness, we randomly sample 15 documents from SciERC and 10 documents from the top 30 most cited papers in ACL Anthology<sup>8</sup> as of June 2023. For ACL papers, we utilize zero-shot Flan-T5-XL to extract keywords. Then we obtain the test set (*FACT-test*) containing 48 instances derived from 25 documents.

The statistics of all datasets are presented in Table 3.

| Datasets          | #Instances |
|-------------------|------------|
| <i>MuP-train</i>  | 18,934     |
| <i>MuP-dev</i>    | 3,604      |
| <i>SciSumm</i>    | 915        |
| <i>Coco-train</i> | 17,865     |
| <i>Coco-dev</i>   | 1,984      |
| <i>MuP-dev-1k</i> | 1000       |
| <i>KW-test</i>    | 229        |
| <i>FACT-test</i>  | 48         |

Table 3: Statistics of the Training Set

## B Evaluation Metrics

For length controllability evaluation, we adopt (1) the Mean of Absolute Deviation (MAD, Liu et al., 2018) of length codes of system-generated summaries and the references, measuring their length distance; and (2) the Pearson Correlation Coefficient (PCC, Liu et al., 2018) between above length codes to access the summary variations as length

<sup>8</sup><https://aclanthology.org/>

signals change. For evaluating keyword controllability, we use (1) the Success Rate (SR, Fan et al., 2018; He et al., 2022)<sup>9</sup> to reflect the fraction of requested keywords actually occurring in the output summaries; and (2) the Factual Correctness (FACT, Krishna et al., 2023) to ensure the generated summaries contain factually accurate information. We introduce the calculation of the evaluation metrics as follows.

**Mean of Absolute Deviation** (MAD, Liu et al., 2018) is used to evaluate the distance between the target length  $L_{target}$  and the length of the system-generated summary  $L_{sys}$ . We categorize summaries into bins based on their lengths: bin 1 contains 0-50 words, bin 2 has 50-100 words, and so forth. We calculate MAD using the bin number of corresponding length with the following equation:

$$MAD = \frac{1}{N} \sum_n^N |L_{sys} - L_{target}|. \quad (1)$$

**Pearson Correlation Coefficient** (PCC, Liu et al., 2018), which is a number between  $-1$  and  $1$  that measures the strength and direction of the relationship between two variables. A number between  $0$  and  $1$  means when one variable changes, the other variable changes in the same direction, and vice versa. For each test instance, we generate five summaries with length signals of 50, 100, 150, 200, and 250 words. Subsequently, we compute the PCC between the actual length and the control signal for each summary.

**Success Rate** (SR, Fan et al., 2018), is the fraction of keywords actually occurring in the output summaries. We calculate SR employing exact matching after stemming.

**Factual Correctness** (FACT, Krishna et al., 2023), is a fine-grained evaluation by human annotators. We ask annotators to make a binary 0/1 judgment for the correctness of each sentence, based on whether it can be entailed or logically inferred from the provided source document. We use voting to decide the final score of the sentence based on judgments of multiple annotators. The FACT score of a batch of generated summaries is the average number of binary scores of all sentences.

We recruit 3 students from our university for annotation, who are majored in computer science and fluent in scientific document reading in English. These annotators are carefully selected based on

their expertise and demonstrated high performance in a trial task, ensuring the quality and reliability of their evaluations. The average annotation time for the three annotators is 18 hours, and we provide reasonable compensation based on their working hours.

## C Experimental Settings

All input source documents are truncated to 1024 tokens to fit in the model. To finetune models, we use a learning rate of  $5e - 4$ , applying StepLR for learning rate scheduling. We conduct all experiments with an RTX 3090 GPU with 24GB memory.

<sup>9</sup>Full article success rate in (He et al., 2022)



# CoLLiE: Collaborative Training of Large Language Models in an Efficient Way

Kai Lv<sup>1\*†</sup>, Shuo Zhang<sup>1\*†</sup>, Tianle Gu<sup>2</sup>, Shuhao Xing<sup>1</sup>, Jiawei Hong<sup>1</sup>, Keyu Chen<sup>1</sup>  
 Xiaoran Liu<sup>1</sup>, Yuqing Yang<sup>1</sup>, Honglin Guo<sup>1</sup>, Tengxiao Liu<sup>1</sup>, Yu Sun<sup>1</sup>  
 Qipeng Guo<sup>1</sup>, Hang Yan<sup>2‡</sup>, Xipeng Qiu<sup>1‡</sup>

<sup>1</sup> School of Computer Science, Fudan University, <sup>2</sup> Shanghai AI Laboratory

<sup>1</sup>{klv21, szhang22, hongjw21, kychen22, liuxr22}@m.fudan.edu.cn

<sup>1</sup>{shxing, hlguo20, qpquo16, xpqiu}@fudan.edu.cn, <sup>2</sup>{gutianle, yanhang}@pjlab.org.cn

## Abstract

Large language models (LLMs) are increasingly pivotal in a wide range of natural language processing tasks. Access to pre-trained models, courtesy of the open-source community, has made it possible to adapt these models to specific applications for enhanced performance. However, the substantial resources required for training these models necessitate efficient solutions. This paper introduces CoLLiE, an efficient library that facilitates collaborative training of large language models using 3D parallelism, parameter-efficient fine-tuning (PEFT) methods, and optimizers such as Lion, Adan, Sophia, and LOMO. With its modular design and comprehensive functionality, CoLLiE offers a balanced blend of efficiency, ease of use, and customization. CoLLiE has proven superior training efficiency in comparison with prevalent solutions in pre-training and fine-tuning scenarios. Furthermore, we provide an empirical evaluation of the correlation between model size and GPU memory consumption under different optimization methods, as well as an analysis of the throughput. Lastly, we carry out a comprehensive comparison of various optimizers and PEFT methods within the instruction-tuning context. CoLLiE is available at <https://github.com/OpenMLLab/collie>.

## 1 Introduction

Large language models (LLMs) have demonstrated remarkable abilities across various natural language processing tasks and showcased potential as intelligent assistants. Thanks to the vibrant open-source community, multiple excellent large language models' weights are accessible, including OPT (Zhang et al., 2022), BLOOM (Scao et al., 2022), LLaMA (Touvron et al., 2023), etc. Despite the impressive general capabilities of pre-trained LLMs, training for particular application scenarios

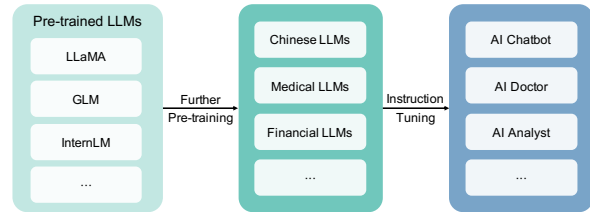


Figure 1: The two stages of training pre-trained language models, during which CoLLiE exhibits efficiency.

can lead to even more outstanding performance. As shown in Figure 1, the training process can be divided into two stages: 1. **Further pre-training**, which supplements specific domain knowledge and expands the vocabulary to enhance tokenization efficiency; 2. **Instruction-tuning**, which adapts the model to downstream tasks and improves its instruction-following ability.

With the scaling of language models, the resources required for training have increased substantially, making it infeasible to train the entire model on a single GPU. Model parallelism addresses this issue by partitioning the model across different GPUs, distributing the training workload among these GPUs. This can be achieved through three methods: tensor parallelism (TP, Shoeybi et al. (2019)), pipeline parallelism (PP, Huang et al. (2019); Narayanan et al. (2019)), and stage 3 of zero redundancy optimizer (ZeRO-3, Rajbhandari et al. (2020)). In addition, during the instruction-tuning stage, there are resource-efficiency and training-effectiveness trade-off approaches (Sun et al., 2023b): parameter-efficient fine-tuning (PEFT) methods (Ding et al., 2023). These methods selectively choose or add a few parameters for training, effectively reducing the GPU memory required to train large language models.

In this context, we introduce CoLLiE, an easy-to-use library for **Collaborative training of Large Language models in an Efficient way**. The library not only integrates the previously mentioned three parallelism strategies and PEFT methods, but also

\*Equal contribution.

†Work done during internship at Shanghai AI Lab.

‡Corresponding authors.

implements efficient optimizers such as Lion (Chen et al., 2023), Adan (Xie et al., 2022), Sophia (Liu et al., 2023), and LOMO (Lv et al., 2023). We have restructured multiple mainstream open-source models to support TP and PP and incorporated FlashAttention (Dao et al., 2022; Dao, 2023) to further boost efficiency, while retaining interfaces similar to HuggingFace models within `CollieModel` class. Training efficiency is one of the most distinctive features of CoLLiE, boasting a significantly higher training throughput compared to current popular solutions. CoLLiE also offers a wide range of functionalities, including data preprocessing, model training, checkpoint saving and monitoring and evaluation during training process. CoLLiE’s modular design allows for flexible combinations of parallelism strategies, PEFT methods, and training hyperparameters, which can be configured simply by modifying the `CollieConfig` class. Furthermore, CoLLiE is purposefully designed with extensibility, providing customizable functionalities. In summary, CoLLiE offers a comprehensive solution that caters to the needs of both beginners and experienced professionals. Our contributions can be summarized as follows:

- We introduce CoLLiE, an efficient and easy-to-use library for collaborative training of large language models.
- We empirically provide the relationship between model size and the actual GPU memory consumption using different optimization methods in real training scenarios.
- We compared the throughput of CoLLiE and the current prevailing solutions in (further) pre-training and fine-tuning scenarios, and CoLLiE demonstrates higher efficiency.
- We conducted a comprehensive comparison of different optimizers and PEFT methods in the context of instruction-tuning.

## 2 Background

**PEFT Methods** There has been a rise in using parameter-efficient fine-tuning (PEFT) techniques to adapt models for instruction-tuning by adjusting partial parameters. One of the early success is adapter tuning (Houlsby et al., 2019), which inserts trainable neural modules into transformers layers while keeping the original model unchanged. In line with adapter tuning, LoRA (Hu et al., 2022)

reparameterizes the dense layers and only updates low rank matrices while introducing no latency during inference. Prefix-tuning (Li and Liang, 2021) trains a task specific prefix prepended to each layer of the transformer encoder and achieves comparable performance with full parameter fine-tuning on generative tasks. Similarly, prompt-tuning (Lester et al., 2021) simplifies the additional prefix to the input embeddings, and only updates the parameters corresponding to the prompts.

While the PEFT library (Mangrulkar et al., 2022) implements these algorithms at the model level, it relies on HuggingFace models and lacks a comprehensive functionality, particularly the necessary integration with model parallelism to facilitate training of extremely large models.

**Parallelism Strategies** Parallelism strategies refer to the methodology of utilizing multiple GPUs to execute training or inference tasks. Data parallelism involves distributing the input data to different GPUs for computation. However, each GPU stores an identical copy of the optimizer state and model weights, which limits the maximum model size that can be trained with data parallelism to that of a single GPU. To mitigate this redundancy, Rajbhandari et al. (2020) proposes a parallelism strategy in the three stages of ZeRO, evenly partitioning the optimizer states, gradients, and weights across different GPUs. Tensor parallelism also partitions the weights evenly, while it varies the approach to partition and communicate. Specifically, whereas ZeRO-3 gathers the weight matrices, tensor parallelism all reduces the intermediate computational results. Pipeline parallelism partitions the model by layers across GPUs, requiring communication only between the layers at the split points. This strategy yields the least communication overhead.

Existing toolkits, such as HuggingFace’s Trainer (Wolf et al., 2020) and LMFlow (Diao et al., 2023), choose ZeRO-3 as parallel method. ZeRO-3 is preferred because it does not impose specific requirements on the model structure, allowing direct usage of HuggingFace models. However, it exhibits lower throughput compared to the combination of TP and PP in scenarios involving large batch size pre-training or constrained communication. CoLLiE supports the hybrid application of data parallelism, tensor parallelism, and pipeline parallelism, collectively termed as 3D parallelism, with the parallel sizes adjustable via `CollieConfig`.

### 3 CoLLiE

In this section, we will introduce the implementation and features of CoLLiE. Figure 2(a) presents an overview of the Collie’s overall structure, centered around the Trainer class. The CollieConfig, CollieModel, CollieDataset, and Optimizer classes serve as inputs to the Trainer. CoLLiE also provides a set of convenient plugins for the Trainer, including Callback, Monitor, Evaluator, and Probe, enabling users to customize the training process. Depending upon the configurations and selected plugins, the Trainer performs the training process, saves model checkpoints, and records system metrics, including loss, throughput, and evaluation results.

As shown in Figure 2(b), based on PyTorch (Paszke et al., 2019) and DeepSpeed (Rasley et al., 2020), CoLLiE employs a collaborative approach using various techniques to facilitate the more efficient training of large language models. Specifically, CoLLiE integrates FlashAttention to enhance efficiency. It implements ZeRO-powered DP, TP, and PP to support 3D parallelism. Additionally, LOMO and PEFT methods are incorporated to realize memory-efficient fine-tuning approaches.

Appendix A provides a brief tour demonstrating how to use CoLLiE for training.

#### 3.1 Collaborative Tuning Methods

##### 3.1.1 3D Parallelism

While distributed training frameworks such as DeepSpeed and Colossal-AI (Bian et al., 2021) support 3D parallelism, models in HuggingFace can only opt for ZeRO-3 for model parallelism due to structural constraints. To fully support 3D parallelism and meet the distributed training needs under different scenarios, CoLLiE rewrites the models using Megatron-LM (Shoeybi et al., 2019) and restructures them according to DeepSpeed’s structure requirements for pipeline models. In the rewriting process, we have maintained the interface to be essentially consistent with the HuggingFace models, and have allowed the direct use of the `from_pretrained` method to load pre-trained models from the HuggingFace hub. This approach significantly reduces the learning curve for users.

##### 3.1.2 Parameter-efficient Fine-tuning

The PEFT library implements state-of-the-art PEFT methods at the model level, but lacks distributed training capabilities. CoLLiE has integrated the

PEFT library into CollieModel, and made necessary patches to enable distributed training.

##### 3.1.3 Efficient Optimizers

In addition to the popular AdamW (Kingma and Ba, 2015) optimizer, several other optimizers have been proposed for the purpose of saving memory, improving optimization results, or accelerating convergence. The implementation of the optimizers in CoLLiE is decoupled from other parts, and incorporates a variety of novel optimizers including Adan, Lion, LOMO, and Sophia. The effectiveness of these optimizers in training large language models is verified and compared in Section 4.3.

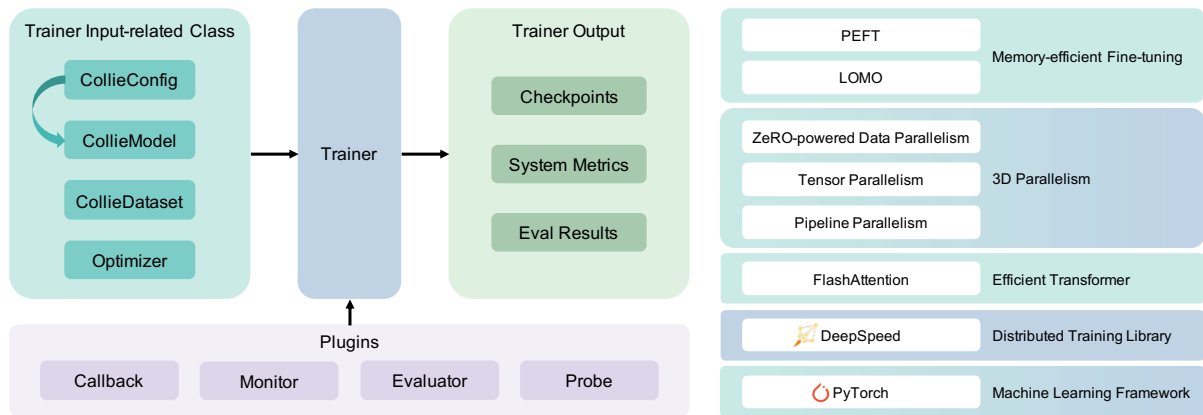
#### 3.2 Models

In addition to the above-mentioned model implementations, CoLLiE has also replaced the naïve self-attention implementation with FlashAttention. Given that FlashAttention has strict requirements regarding hardware and CUDA versions, for users without newer training equipment, we have added the ‘`use_flash`’ option to the CollieConfig to allow for one-click disabling of FlashAttention usage. Currently, CoLLiE has implemented a variety of language models, including but not limited to LLaMA, InternLM (Team, 2023), ChatGLM (Du et al., 2022), and MOSS (Sun et al., 2023a), with the intention to support more models in the future.

#### 3.3 Configuration

CoLLiE offers a unified class, CollieConfig, to manage configurations including model config, parallelism strategy, DeepSpeed configuration, PEFT configuration, and training hyperparameters. Based on the contents of CollieConfig, CollieModel will automatically adjust the partitioning of model parameters and the structure of the model, and the Trainer will modify the training process. Through CollieConfig, users can conveniently combine different pre-trained language models, fine-tuning methods, and hyperparameters.

Model config refers to parameters that describe the model structure, such as `hidden_size`, `num_attention_heads`, and `num_hidden_layers`. The model config is fixed for pre-trained language models, and we provide a `from_pretrained` interface, identical to HuggingFace’s, to initialize model config. Users can also specify the model config to customize their models, intended for training from scratch without the use of pre-trained models. Below is a code



(a) Architecture of CoLLiE. The blocks represent different modularly designed classes or the outputs of the Trainer. (b) Features of CoLLiE. CoLLiE supports a collaborative suite of high-efficiency optimization features.

Figure 2: Overall architecture and features of CoLLiE. Features in (b) are color-coded to match corresponding part in (a), indicating where each feature is implemented.

example of downloading the model config from the HuggingFace hub, initializing the CollieConfig, and setting up to use FlashAttention.

```
config = CollieConfig.from_pretrained(
    'meta-llama/Llama-2-7b-hf'
)
config.use_flash = True
```

CollieConfig streamlines the setup for 3D parallelism as followings. CoLLiE will automatically configure the distributed environment and partition the parameters according, relieving users from managing the complexities of distributed training. The number of GPUs required for training is equal to the product of the three parallelism sizes.

```
config.dp_size = 1
config.tp_size = 8
config.pp_size = 2
```

CoLLiE implements distributed training based on DeepSpeed, and DeepSpeed-related configurations can be set via ds\_config. The configurations related to PEFT methods can also be set via peft\_config. Below is an example for mixed-precision training with FP16 and LoRA.

```
config.ds_config = {
    'fp16': {'enabled': True}
}
config.peft_config = LoraConfig(
    r=4,
    lora_alpha=32,
    target_modules=['q_proj', 'v_proj'],
    bias='none',
    task_type='CAUSAL_LM'
)
```

The training hyperparameters can also be configured through CollieConfig. Loading

CollieConfig from a file is supported and we provide a convenient Command Line Interface (CLI) to generate the required configuration file.

### 3.4 Dataset

To facilitate data processing, CoLLiE provides three Dataset classes for training, evaluation of generation tasks, and evaluation of classification tasks respectively: CollieDatasetForTraining, CollieDatasetForGeneration, CollieDatasetForClassification. These three classes can either read data from a JSON file or a list of dictionaries, process it, and store the results on disk for direct reading next time.

CollieDatasetForTraining accepts two forms of input, one with the field “text”, and the other with fields “input” and “output”. The loss of tokens in the field “text” or “output” will be computed, corresponding to pre-training and instruction-tuning tasks, respectively. CollieDatasetForGeneration and CollieDatasetForClassification both inherit from the CollieDatasetForTraining class, serving as the datasets for generation tasks and classification tasks, respectively. The CollieDatasetForGeneration can accept “text” as a required field and “target” as an optional field. The model generates output based on the “text” and the “target” is used to compute metrics in Evaluator. On the other hand, CollieDatasetForClassification can accept “input”, “output”, and “target” fields. The “input” represents the question, “output” includes all possible options, and “target” indicates which

option should be chosen.

### 3.5 Controller

In this section, we will introduce three modularly designed classes centered around Trainer. Trainer calls the Evaluator and Server classes unidirectionally to serve the purposes of evaluation or manual probing of the model during training.

#### 3.5.1 Trainer

Distributed training, including the initialization of the distributed environment, training loop, and the saving of model weights and checkpointing, can be complex. CoLLiE provides a Trainer to alleviate this burden on users. The Trainer wraps the relatively fixed training loop and offers multiple interfaces for users to further customize the training process. These include the `train_fn` function that obtains output based on a given batch of input and the `loss_fn` function that obtains loss based on the batch and output from `train_fn`. Moreover, CoLLiE offers several plugins to enrich functionality.

**Monitor** The Monitor class tracks various metrics such as loss, learning rate, throughput, and memory usage during the training process, and records them to Tensorboard, WandB, or local CSV files.

**Callback** The Callback class can be invoked at various callback points during the training process, allowing users to customize the training loop. CoLLiE has implemented callbacks that save model weights and training checkpoints when necessary, or load the model weights of the best evaluated results after training. These callbacks are all implemented based on the same base class. Users can inherit from this base class and override different methods to choose the callback timing and actions.

#### 3.5.2 Evaluator

The Evaluator class is used in conjunction with the Metric class for assessing model performance. We implemented three types of Evaluator, intended for generation tasks, classification tasks, and perplexity assessment, by subclassing the Evaluator base class and overriding the `eval_fn` method. The return value of the `eval_fn` method in the Evaluator is accepted as input by the update function of the Metric class. The Metric class's update method updates the variables necessary for calculating the metric after processing each batch from the evaluation dataset. After the evaluation dataset is fully processed, the `get_metric` function is employed to compute the metric. The

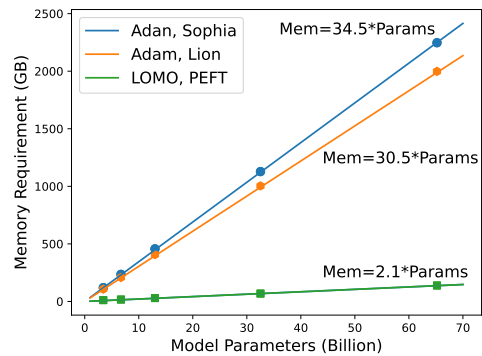


Figure 3: Memory requirements when training models with different parameters under various configurations.

Evaluator class can either be provided to the Trainer for assessment during the training process or it can evaluate the model independently without the dependency on the Trainer.

#### 3.5.3 Server

The Server class offers web-based, interactive and streaming generated sequences feature, enabling users to conveniently deploy trained models for web-based use, as well as manually probe model performance during training. The DataProvider class supplies asynchronous inference data for Server as a subprocess. When the Server is integrated into the Trainer, users can input prompts via the web interface. Once the current batch training is completed, an output will be generated based on the user's input prompt and returned to the web interface for user's review.

### 3.6 Documentation

We provide API documentation and easily understandable tutorials<sup>1</sup> for users who are new to CoLLiE and distributed training. Comprehensive code examples<sup>2</sup> including vocabulary expansion, instruction-tuning, and downstream tasks such as summary and translation are also available.

## 4 Evaluation

### 4.1 Memory Requirement

While Rajbhandari et al. (2020) estimates the total GPU memory required for model training as 18 times the number of model parameters in bytes, more GPU memory is consumed in reality. This is because this estimation only considers memory used by parameters, gradients, and the optimizer

<sup>1</sup>[https://openmlab-collie.readthedocs.io/zh\\_CN/latest](https://openmlab-collie.readthedocs.io/zh_CN/latest)

<sup>2</sup><https://github.com/OpenMLLab/collie/tree/main/examples>

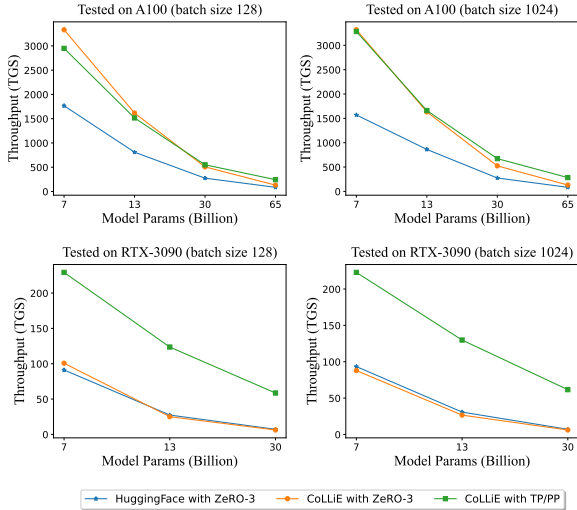


Figure 4: Throughput tested on A100 and RTX-3090.

states, and neglects other components such as activation values and buffers used for communication.

In this section, we profile the actual memory requirements for training models under different configurations to facilitate users in more accurately estimating the model size that their devices can train. As depicted in Figure 3, the most commonly used Adam optimizer requires 30.5 times the amount of memory relative to the model parameters, which is consistent with Lion. Adan and Sophia optimizers use 4 times more memory for intermediate variables when updating parameters, amounting to 34.5 times the parameter size. The LOMO optimizer, without storing any optimizer state or gradient, only requires 2.1 times the parameter size in memory, almost all of which is consumed by the half-precision parameters. PEFT methods, which update only a small proportion of parameters, have a memory usage similar to LOMO.

## 4.2 Throughput Analyses

We take HuggingFace models with ZeRO-3 as a baseline to analyse the throughput of CoLLiE during pre-training (with batch size of 1024) and fine-tuning (with batch size of 128). The corpus we used consists of the first 10,000 entries from the Pile (Gao et al., 2021). The throughput is measured by the number of tokens processed by each GPU per second, referred to as TGS.

As shown in Figure 4, on the A100 connected by NVLink, CoLLiE’s throughput significantly surpasses the baseline attributed to the integration of FlashAttention. On the RTX-3090, where communication is limited by PCIe, CoLLiE achieves

|         | MMLU        | BBH         | GSM         | HumanEval   | AlpacaFarm  | Avg.        |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|
| Vanilla | 62.4        | 56.9        | 53.9        | 20.7        | 4.7         | 39.7        |
| LoRA    | 62.7        | <b>58.7</b> | <b>60.5</b> | <b>32.9</b> | 69.6        | <b>56.9</b> |
| LOMO    | 62.1        | 56.9        | 57.6        | 28.1        | 65.2        | 54.0        |
| Lion    | 58.2        | 52.6        | 41.3        | 25.0        | 66.2        | 48.7        |
| Adan    | 57.3        | 51.9        | 37.3        | 21.3        | 62.5        | 46.1        |
| Adam    | <b>63.0</b> | 58.0        | 55.3        | 28.1        | <b>73.1</b> | 55.5        |

Table 1: Comparison of different training methods on GPT4-Alpaca. Instruction-tuning significantly enhances the instruction-following ability of vanilla LLaMA-65B.

substantially higher throughput by a more appropriate parallelism approach, namely TP and PP.

## 4.3 Empirical Assessment of Effectiveness

We employ various training methods on GPT-4-Alpaca (Peng et al., 2023) for the LLaMA-65B model and evaluate the factual knowledge, reasoning abilities, code capabilities, and instruction-following abilities using MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and AlpacaFarm (Dubois et al., 2023). The hyperparameters and templates for training and evaluating can be found in Appendix B.3 and Appendix C, respectively.

The results in Table 1 demonstrate that while the vanilla LLaMA-65B already exhibits substantial capabilities, it struggles to effectively follow instructions from actual users. The performance of the models significantly improves on average after instruction-tuning. Training methods such as LoRA, LOMO, and AdamW significantly enhance the model’s ability to follow instructions without compromising its other performance.

## 5 Conclusion

We have introduced CoLLiE, a library for collaboratively training large language models in an efficient way. CoLLiE offers efficient models with FlashAttention and structurally supportive for 3D parallelism. Moreover, CoLLiE provides a comprehensive and customizable Trainer to assist users throughout the training process, supporting various training methods. We have tested the relationship between the GPU memory requirements and model parameter sizes as a reference for users. In terms of throughput, CoLLiE is significantly more efficient than HuggingFace’s parallel solutions. The effectiveness of different training methods are also empirically assessed on instruction-tuning tasks.



## Limitations

We discuss the limitations of this paper from the following two aspects:

1) Although we profile the memory usage under real training conditions in this paper, a more fine-grained memory allocation situation is not provided. In the future, we plan to develop a fine-grained memory monitor to assist users in training.

2) Due to resource and time constraints, this paper only presents the instruction-tuning results of LLaMA-65B with different training methods. This restricts users from comparing the performance of models of different sizes. We will provide performances of more models under various training methods and continuously update them on our Github repository for user reference. Furthermore, while CoLLiE has implemented the Sophia optimizer to enhance pre-training efficiency, we have not conducted extensive experiments under costly pre-training tasks.

## Acknowledgements

This work was supported by the National Key Research and Development Program of China (No.2022ZD0160102) and National Natural Science Foundation of China (No.62022027).

## References

- Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. 2021. [Colossal-ai: A unified deep learning system for large-scale parallel training](#). *CoRR*, abs/2110.14883.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *CoRR*, abs/2107.03374.
- Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. [Symbolic discovery of optimization algorithms](#). *CoRR*, abs/2302.06675.
- 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](#). *CoRR*, abs/2110.14168.
- Tri Dao. 2023. [Flashattention-2: Faster attention with better parallelism and work partitioning](#). *CoRR*, abs/2307.08691.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. [Flashattention: Fast and memory-efficient exact attention with io-awareness](#). In *NeurIPS*.
- Shizhe Diao, Rui Pan, Hanze Dong, Kashun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. [Lmflow: An extensible toolkit for finetuning and inference of large foundation models](#). *CoRR*, abs/2306.12420.
- Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. [Parameter-efficient fine-tuning of large-scale pre-trained language models](#). *Nat. Mac. Intell.*, 5(3):220–235.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. [Glm: General language model pretraining with autoregressive blank infilling](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.
- Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [Alpaca-farm: A simulation framework for methods that learn from human feedback](#). *CoRR*, abs/2305.14387.
- 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](#). *CoRR*, abs/2101.00027.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

- 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. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, Hyoungho Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. [Gpipe: Efficient training of giant neural networks using pipeline parallelism](#). 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 103–112.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- 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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.
- Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2023. [Sophia: A scalable stochastic second-order optimizer for language model pre-training](#). *CoRR*, abs/2305.14342.
- Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. 2023. [Full parameter fine-tuning for large language models with limited resources](#). *CoRR*, abs/2306.09782.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>.
- Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. [Pipedream: generalized pipeline parallelism for DNN training](#). In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pages 1–15. ACM.
- 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.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. [Instruction tuning with GPT-4](#). *CoRR*, abs/2304.03277.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. [Zero: memory optimizations toward training trillion parameter models](#). In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. [Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters](#). In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie 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.
- 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.
- Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejiang Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, and Xipeng Qiu. 2023a. Moss: Training conversational language models from synthetic data. <https://github.com/OpenMLab/MOSS>.
- Xianghui Sun, Yunjie Ji, Baochang Ma, and Xianggang Li. 2023b. A comparative study between full-parameter and lora-based fine-tuning on chinese instruction data for instruction following large language model. *CoRR*, abs/2304.08109.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.
- InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. <https://github.com/InternLM/InternLM>.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.
- Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *CoRR*, abs/2306.04751.
- 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.
- Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. 2022. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *CoRR*, abs/2208.06677.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. 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. *CoRR*, abs/2205.01068.

## A Code Example

Listing 1 presents the simplest code example for training with CoLLiE.

## B Hyperparameters

| Training Methods | LR   | Batch Size | Weight Decay | Epochs |
|------------------|------|------------|--------------|--------|
| LoRA             | 3e-4 | 128        | 1e-2         | 3      |
| LOMO             | 1e-2 | 16         | -            | 5      |
| Lion             | 3e-6 | 128        | 3e-2         | 3      |
| Adan             | 5e-5 | 128        | 2e-2         | 3      |
| Adam             | 1e-5 | 128        | 1e-2         | 3      |

Table 2: Hyperparameters for training.

### B.1 Memory Requirements

We choose the combination of Tensor Parallelism (TP) and Pipeline Parallelism (PP) as our parallelism strategy. The batch size is set to 2048, and the gradient accumulation steps are set to 2. It’s worth noting that increasing the value of the gradient accumulation steps would not significantly increase the memory usage.

### B.2 Throughput

In our throughput tests, we consistently employ Adam as the optimizer. We utilize the default settings of DeepSpeed for ZeRO-3 and strive to maximize the micro batch size to enhance throughput. For Tensor Parallelism/Pipeline Parallelism (TP/PP), we ensure that the gradient accumulation steps are more than four times the number of pipeline stages to minimize the bubble. The specific configurations are illustrated in Table 4.

### B.3 Instruction-tuning

As shown in Table 2, we have adopted the learning rates and batch sizes from the Tulu (Wang et al., 2023) and Alpaca-LoRA projects<sup>3</sup> for AdamW and LoRA. To achieve better performance for LoRA, we have replaced all modules with LoRA layers, not just the q-v module. For Lion and Adan, we have used the learning rates recommended in the paper. Specifically, the learning rate for Lion is 3-10 times smaller than that of AdamW, with the weight decay correspondingly 3-10 times larger. The learning rate for the Adan optimizer is 5-10

<sup>3</sup><https://github.com/tloen/alpaca-lora>

---

### Template for entries with input

---

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

```
### Instruction:
{instruction}
```

```
### Input:
{input}
```

```
### Response: {response}
```

---

### Template for entries without input

---

Below is an instruction that describes a task. Write a response that appropriately completes the request.

```
### Instruction:
{instruction}
```

```
### Response: {response}
```

---

Table 3: Templates used for training.

times larger than that of AdamW, with a weight decay of 0.02. For the LOMO optimizer, which is similar to SGD, we have utilized a larger learning rate and a smaller batch size.

## C Templates

### C.1 Alpaca

We follow the template provided by the Alpaca repository<sup>4</sup> for training, as shown in Table 3.

### C.2 Evaluation

We modify the evaluation template based on the template used during training, as shown in Table 5. The template used for evaluate on AlpacaFarm is identical to that of training on Alpaca.

<sup>4</sup>[https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca)

Listing 1: An example for training with CoLLiE.

```

import torch
from collie.config import CollieConfig
from collie.models import LlamaForCausalLM
from collie.controller import Trainer
model_name_or_path = 'meta-llama/Llama-2-7b-hf'
# load model config from huggingface hub
config = CollieConfig.from_pretrained(model_name_or_path)
# set pipeline parallelism size via config
config.pp_size = 8
# load pre-trained weights from huggingface hub
# and partition the weights into 8 stages for pipeline parallelism
model = LlamaForCausalLM.from_pretrained(
    model_name_or_path,
    config=config
)
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)
# one of the two formats collie defined for training
train_dataset = [
    {'text': 'Collie is a package for training large language models.'}
    for _ in range(100)
]
trainer = Trainer(
    model=model,
    optimizer=optimizer,
    config=config,
    train_dataset=train_dataset,
)
# start the training process
trainer.train()

```

| Model Params            |                 | 7B                    | 13B            | 30B            | 65B            |
|-------------------------|-----------------|-----------------------|----------------|----------------|----------------|
| Device Mode             |                 | A100                  |                |                |                |
|                         |                 | Fine-tune / Pre-train |                |                |                |
| # GPU                   |                 | 4                     | 8              | 16             | 32             |
| HuggingFace with ZeRO-3 | Batch Size, GAS | 64,2 / 64,16          | 64,2 / 64,16   | 64, 2 / 128,8  | 128,1 / 128,8  |
| CoLLiE with ZeRO-3      | Batch Size, GAS | 64,2 / 64,16          | 128,1 / 64,16  | 128,1 / 128,8  | 128,1 / 128,8  |
| CoLLiE with TP/PP       | Batch Size, GAS | 4,32 / 8,128          | 2,64 / 16,64   | 1,128 / 2,512  | 1,128 / 1,1024 |
| Device Mode             |                 | RTX-3090              |                |                |                |
|                         |                 | Fine-tune / Pre-train |                |                |                |
| # GPU                   |                 | 8                     | 24             | 48             | -              |
| HuggingFace with ZeRO-3 | Batch Size, GAS | 8,16 / 8,128          | 24,6 / 24,43   | 48,3 / 48,22   | -              |
| CoLLiE with ZeRO-3      | Batch Size, GAS | 8,16 / 8,128          | 24,6 / 24,43   | 48,3 / 48,22   | -              |
| CoLLiE with TP/PP       | Batch Size, GAS | 1,128 / 1,1024        | 1,128 / 1,1024 | 1,128 / 1,1024 | -              |

Table 4: **Hyperparameters for testing throughput.** We report the number of model parameters (Model Params), device (Device), mode (Mode) and number of GPU (#GPU). We also report the corresponding batch size (Batch Size) and GAS (Gradient Accumulation Steps) for HuggingFace with ZeRO-3, CoLLiE with ZeRO-3 and CoLLiE with TP/PP.

---

**MMLU**

---

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

**### Instruction:**

The following is a multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

**### Input:**

Find all  $c$  in  $\mathbb{Z}_3$  such that  $\mathbb{Z}_3[x]/(x^2 + c)$  is a field.

- A. 0
- B. 1
- C. 2
- D. 3

**### Response:** B

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

**### Instruction:**

The following is multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

**### Input:**

Statement 1 | If  $aH$  is an element of a factor group, then  $|aH|$  divides  $|a|$ . Statement 2 | If  $H$  and  $K$  are subgroups of  $G$  then  $HK$  is a subgroup of  $G$ .

- A. True, True
- B. False, False
- C. True, False
- D. False, True

**### Response:** B

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

**### Instruction:**

The following is multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

**### Input:**

Statement 1 | Every element of a group generates a cyclic subgroup of the group. Statement 2 | The symmetric group  $S_{10}$  has 10 elements.

- A. True, True
- B. False, False
- C. True, False
- D. False, True

**### Response:** C

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

**### Instruction:**

The following is multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

### Input:

Statement 1 | Every function from a finite set onto itself must be one to one. Statement 2 | Every subgroup of an abelian group is abelian.

- A. True, True
- B. False, False
- C. True, False
- D. False, True

### Response: A

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

The following is multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

### Input:

Find the characteristic of the ring  $2\mathbb{Z}$ .

- A. 0
- B. 3
- C. 12
- D. 30

### Response: A

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

The following is multiple choice question (with answers) about abstract algebra. You need to answer the question by selecting the correct option.

### Input:

Input

### Response:

---

**BBH**

---

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Evaluate the result of a random Boolean expression.

### Input:

not ( ( not not True ) ) is

### Response: Let's think step by step.

Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.

We first simplify this expression "Z" as follows: " $Z = \text{not} ( ( \text{not not True} ) ) = \text{not} ( ( A ) )$ " where " $A = \text{not not True}$ ".

Let's evaluate A:  $A = \text{not not True} = \text{not} ( \text{not True} ) = \text{not False} = \text{True}$ .

Plugging in A, we get:  $Z = \text{not} ( ( A ) ) = \text{not} ( ( \text{True} ) ) = \text{not True} = \text{False}$ . So the answer is False.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:  
Evaluate the result of a random Boolean expression.

### Input:  
True and False and not True and True is

### Response: Let's think step by step.  
Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.  
We first simplify this expression "Z" as follows: "Z = True and False and not True and True = A and B" where "A = True and False" and "B = not True and True".  
Let's evaluate A: A = True and False = False.  
Let's evaluate B: B = not True and True = not (True and True) = not (True) = False.  
Plugging in A and B, we get: Z = A and B = False and False = False. So the answer is False.  
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:  
Evaluate the result of a random Boolean expression.

### Input:  
not not ( not ( False ) ) is

### Response: Let's think step by step.  
Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.  
We first simplify this expression "Z" as follows: "Z = not not ( not ( False ) ) = not not ( A )" where "A = not ( False )".  
Let's evaluate A: A = not ( False ) = not False = True.  
Plugging in A, we get: Z = not not ( A ) = not not (True) = not not False = True. So the answer is True.  
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:  
Evaluate the result of a random Boolean expression.

### Input:

{input}

### Response: Let's think step by step.  
Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.

---

### GSM8K

---

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:  
Given a problem scenario with numerical data, perform the necessary calculations and provide a detailed step-by-step solution, ending the response with 'The answer is'.

### Input:  
Angelo and Melanie want to plan how many hours over the next week they should study together for their test next week. They have 2 chapters of their textbook to study and 4 worksheets to memorize. They figure out that they should dedicate 3 hours to each chapter of their textbook and 1.5 hours for each worksheet. If they plan to study no more than 4 hours each day, how many days should they plan to study total over the next week if they take a 10-minute break every hour, include 3 10-minute snack breaks each day, and 30 minutes for lunch each day?



### Response: Let's think step by step. Angelo and Melanie think they should dedicate 3 hours to each of the 2 chapters, 3 hours x 2 chapters = 6 hours total. For the worksheets they plan to dedicate 1.5 hours for each worksheet, 1.5 hours x 4 worksheets = 6 hours total. Angelo and Melanie need to start with planning 12 hours to study, at 4 hours a day, 12 / 4 = 3 days. However, they need to include time for breaks and lunch. Every hour they want to include a 10-minute break, so 12 total hours x 10 minutes = 120 extra minutes for breaks. They also want to include 3 10-minute snack breaks, 3 x 10 minutes = 30 minutes. And they want to include 30 minutes for lunch each day, so 120 minutes for breaks + 30 minutes for snack breaks + 30 minutes for lunch = 180 minutes, or 180 / 60 minutes per hour = 3 extra hours. So Angelo and Melanie want to plan 12 hours to study + 3 hours of breaks = 15 hours total. They want to study no more than 4 hours each day, 15 hours / 4 hours each day = 3.75 They will need to plan to study 4 days to allow for all the time they need. The answer is 4

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Given a problem scenario with numerical data, perform the necessary calculations and provide a detailed step-by-step solution, ending the response with 'The answer is'.

### Input:

Mark's basketball team scores 25 2 pointers, 8 3 pointers and 10 free throws. Their opponents score double the 2 pointers but half the 3 pointers and free throws. What's the total number of points scored by both teams added together?

### Response: Let's think step by step. Mark's team scores 25 2 pointers, meaning they scored  $25 * 2 = 50$  points in 2 pointers. His team also scores 8 3 pointers, meaning they scored  $8 * 3 = 24$  points in 3 pointers They scored 10 free throws, and free throws count as one point so they scored  $10 * 1 = 10$  points in free throws. All together his team scored  $50 + 24 + 10 = 84$  points Mark's opponents scored double his team's number of 2 pointers, meaning they scored  $50 * 2 = 100$  points in 2 pointers. His opponents scored half his team's number of 3 pointers, meaning they scored  $24 / 2 = 12$  points in 3 pointers. They also scored half Mark's team's points in free throws, meaning they scored  $10 / 2 = 5$  points in free throws. All together Mark's opponents scored  $100 + 12 + 5 = 117$  points The total score for the game is both team's scores added together, so it is  $84 + 117 = 201$  points The answer is 201

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Given a problem scenario with numerical data, perform the necessary calculations and provide a detailed step-by-step solution, ending the response with 'The answer is'.

### Input:

Bella has two times as many marbles as frisbees. She also has 20 more frisbees than deck cards. If she buys  $\frac{2}{5}$  times more of each item, what would be the total number of the items she will have if she currently has 60 marbles?

### Response: Let's think step by step. When Bella buys  $\frac{2}{5}$  times more marbles, she'll have increased the number of marbles by  $\frac{2}{5} * 60 = 24$  The total number of marbles she'll have is  $60 + 24 = 84$  If Bella currently has 60 marbles, and she has two times as many marbles as frisbees, she has  $60 / 2 = 30$  frisbees. If Bella buys  $\frac{2}{5}$  times more frisbees, she'll have  $\frac{2}{5} * 30 = 12$  more frisbees. The total number of frisbees she'll have will increase to  $30 + 12 = 42$  Bella also has 20 more frisbees than deck cards, meaning she has  $30 - 20 = 10$  deck cards If she buys  $\frac{2}{5}$  times more deck cards, she'll have  $\frac{2}{5} * 10 = 4$  more deck cards. The total number of deck cards she'll have is  $10 + 4 = 14$  Together, Bella will have a total of  $14 + 42 + 84 = 140$  items The answer is 140

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Given a problem scenario with numerical data, perform the necessary calculations and provide a detailed step-by-step solution, ending the response with 'The answer is'.

### Input:

A group of 4 fruit baskets contains 9 apples, 15 oranges, and 14 bananas in the first three baskets and 2 less of each fruit in the fourth basket. How many fruits are there?

### Response: Let's think step by step. For the first three baskets, the number of apples and oranges in one basket is  $9+15=24$  In total, together with bananas, the number of fruits in one basket is  $24+14=38$  for the first three baskets. Since there are three baskets each having 38 fruits, there are  $3*38=114$  fruits in the first three baskets. The number of apples in the fourth basket is  $9-2=7$  There are also  $15-2=13$  oranges in the fourth basket The combined number of oranges and apples in the fourth basket is  $13+7=20$  The fourth basket also contains  $14-2=12$  bananas. In total, the fourth basket has  $20+12=32$  fruits. The four baskets together have  $32+114=146$  fruits. The answer is 146

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Given a problem scenario with numerical data, perform the necessary calculations and provide a detailed step-by-step solution, ending the response with 'The answer is'.

### Input:

{question}

### Response: Let's think step by step.

---

### HumanEval

---

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Complete the following python code.

### Input:

Check if in given list of numbers, are any two numbers closer to each other than given threshold.

```
>>> has_close_elements([1.0, 2.0, 3.0], 0.5) False
```

```
>>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3) True
```

### Response:

```
from typing import List
```

```
def has_close_elements(numbers: List[float], threshold: float) -> bool:
```

```
    """ Check if in given list of numbers, are any two numbers closer to each other than given threshold.
```

```
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5) False
```

```
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3) True """
```

---

Table 5: The templates used for evaluation.



# Video-LLaMA

## An Instruction-tuned Audio-Visual Language Model for Video Understanding

Hang Zhang<sup>1 2</sup>    Xin Li<sup>1 2\*</sup>    Lidong Bing<sup>1 2</sup>

<sup>1</sup> DAMO Academy, Alibaba Group

<sup>2</sup> Hupan Lab, 310023, Hangzhou, China

{zh401075, xinting.lx, l.bing}@alibaba-inc.com

### Abstract

We present Video-LLaMA<sup>1</sup> a multi-modal framework that empowers Large Language Models (LLMs) with the capability of understanding both visual and auditory content in the video. Video-LLaMA bootstraps cross-modal training from the frozen pre-trained visual & audio encoders and the frozen LLMs. Unlike previous works that complement LLMs to process the visual or audio signals only (Zhu et al., 2023; Liu et al., 2023; Huang et al., 2023a), Video-LLaMA enables video comprehension by tackling two challenges: (1) capturing the temporal changes in visual scenes, (2) integrating audio-visual signals. To counter the first challenge, we propose a Video Q-former to assemble a pre-trained image encoder into our video encoder and introduce a video-to-text generation task to learn video-language correspondence. For the second challenge, we leverage ImageBind (Girdhar et al., 2023), a universal embedding model aligning multiple modalities, as the pre-trained audio encoder and introduce an Audio Q-former on top of ImageBind to learn reasonable auditory query embeddings for the LLM module. To align the output of both visual & audio encoders with LLM’s embedding space, we first train Video-LLaMA on massive video/image-caption pairs and then tune our model with visual-instruction datasets of moderate amount but higher quality. We found Video-LLaMA shows the ability to perceive and comprehend video content and generate meaningful responses grounded in the visual and auditory information presented in the videos.

### 1 Introduction

Large Language Models (LLMs) (Chowdhery et al., 2022; Bai et al., 2022; OpenAI, 2023) have demonstrated remarkable capability of understanding and

following user intentions and instructions<sup>234</sup>. Typically, the user requests and the corresponding responses from LLMs are all in texts, however, text-only human-computer interaction is not sufficient for many application scenarios because real-world information is usually multi-modal. In order to further explore the potential of LLMs, many researchers attempt to endow LLMs with the capability of understanding multi-modal content (Huang et al., 2023a; Zhang et al., 2023b; Yin et al., 2023).

Among these efforts, Alayrac et al. (2022b); Wang et al. (2022); Huang et al. (2023b); Xu et al. (2023b); Zhang et al. (2023b); Sun et al. (2023) pre-train multi-modal LLMs with massive interleaved image-text data or speech-text data to accommodate multi-modal input. Meanwhile, another group of works adopts a more parameter-efficient way by complementing LLMs with off-the-shelf vision or speech foundation models to achieve multi-modal understanding (Li et al., 2023b; Zhu et al., 2023; Liu et al., 2023; Ye et al., 2023; Zhang et al., 2023a; Huang et al., 2023a; Wu et al., 2023b; Su et al., 2023; Li et al., 2023a).

Despite their effectiveness, these approaches are dedicated to aligning the input from exactly one additional modality with text (i.e., image or audio), which is unsatisfactory for video understanding. Concretely, empowering LLMs to understand video requires comprehensive processing for different modalities including visual input, auditory input, and textual output, which is more challenging than image-only understanding and audio-only understanding tasks. Although there are several recent works attempt to unleash the video understanding capability of LLMs (Li et al., 2023c; Maaz et al., 2023; Luo et al., 2023), their primary objective is to comprehend only the visual content of the video, with the auditory content remaining unused.

\*Xin Li is the corresponding author.

<sup>1</sup>The video demonstration is available at <https://youtu.be/RDNYs3Rswhc>

<sup>2</sup><https://chat.openai.com/chat>

<sup>3</sup><https://www.anthropic.com/product>

<sup>4</sup><https://bard.google.com/>

| Model Name                        | Ability      |              |       |
|-----------------------------------|--------------|--------------|-------|
|                                   | Static Image | Silent Video | Audio |
| BLIP2 (Li et al., 2023b)          | ✓            |              |       |
| MiniGPT4 (Zhu et al., 2023)       | ✓            |              |       |
| LLaVA (Liu et al., 2023)          | ✓            |              |       |
| mPLUG-Owl (Ye et al., 2023)       | ✓            | ✓            |       |
| VideoChat (Li et al., 2023c)      | ✓            | ✓            |       |
| AudioGPT (Huang et al., 2023a)    |              |              | ✓     |
| Video-ChatGPT (Maaz et al., 2023) | ✓            | ✓            |       |
| Video-LLaMA                       | ✓            | ✓            | ✓     |

Table 1: Comparison with popular multi-modal large language models. Video-LLaMA has the unique ability to comprehend auditory and visual information simultaneously.

In this work, to fill in the blank of audio-visual LLMs, we investigate the possibility of building multi-modal LLMs that support the input of video and allow users to chat with computers around the user-uploaded video, which is usually composed of multiple video frames and audio. Instead of employing external perception models to convert visual/auditory signals to textual signals (Shen et al., 2023; Li et al., 2023c), we choose to build an end-to-end model that can handle the data from multiple modalities within one single framework. Specifically, we adopt the idea of BLIP-2 (Li et al., 2023b) to guarantee the efficiency of cross-modal pre-training. To explicitly capture the change of visual scenes in the video, we use a pre-trained visual encoder to separately compute frame representations. Then, we introduce a frame embedding layer to inject temporal information and a video Q-Former to generate visual query tokens. As for the audio signals from the video, we additionally leverage a pre-trained audio encoder as well as an audio Q-former to learn reasonable auditory query embeddings (see the right part of Figure 1).

To align textual output with video, we devise multi-branch cross-modal pre-training to learn the vision-language correspondence and the audio-language correspondence. For vision-language correspondence, we first pre-train the vision-related components on a large-scale video caption dataset with a video-clips-to-text generation task. To enhance the understanding of static visual concepts, we also add image-caption data into this pre-training stage. Then, we further fine-tune these components on a video-based conversation dataset to execute visual instruction tuning. For the alignment between the audio encoder and language decoder, we further pre-train the audio-related components on an audio caption dataset with an audio-

to-text generation task. For the audio-language correspondence, we leverage Imagebind (Girdhar et al., 2023) as an encoder, which performs exceptionally well in aligning different modalities to a common embedding space. Given the limited availability of audio-text data, we also utilize vision-text data to train the audio-related components. These components learn to align the common embedding space provided by Imagebind with the embedding space of LLMs. Despite not being explicitly trained with audio-text data, Video-LLaMA exhibits a remarkable zero-shot audio understanding capability during inference.

As shown in Table 1, our Video-LLaMA stands out from other existing multi-modal LLMs in terms of its distinctively comprehensive comprehension of audiovisual modal information in videos. In summary, our contributions are as follows:

- We propose Video-LLaMA, a multi-modal framework that enables LLM to simultaneously process both the visual and auditory content of a given video and engage in conversation with humans.
- To empower LLMs with video understanding capability, we propose a multi-branch cross-modal pre-training framework to achieve both vision-language alignment and audio-language alignment.
- We open-source the entire codebase for pre-training and fine-tuning as well as the model weights of all the variants of Video-LLaMA<sup>5</sup>. We also prepared the demos for video-grounded conversation<sup>6,7</sup>.

## 2 Method

Video-LLaMA aims to empower frozen LLMs with the capability of understanding both visual and auditory content in videos. As shown in Figure 1, we design two branches, namely Vision-Language Branch and Audio-Language Branch, to respectively transform the video frames and audio signals into query representations that are compatible with the textual inputs of LLMs. In this section, we first introduce the overall architecture and the building blocks of each branch. Then, we delineate the procedures of the proposed multi-branch cross-modal pre-training and audio-visual instruction tuning.

<sup>5</sup><https://github.com/DAMO-NLP-SG/Video-LLaMA>

<sup>6</sup><https://huggingface.co/spaces/DAMO-NLP-SG/Video-LLaMA>

<sup>7</sup><https://modelscope.cn/studios/damo/video-llama/summary>

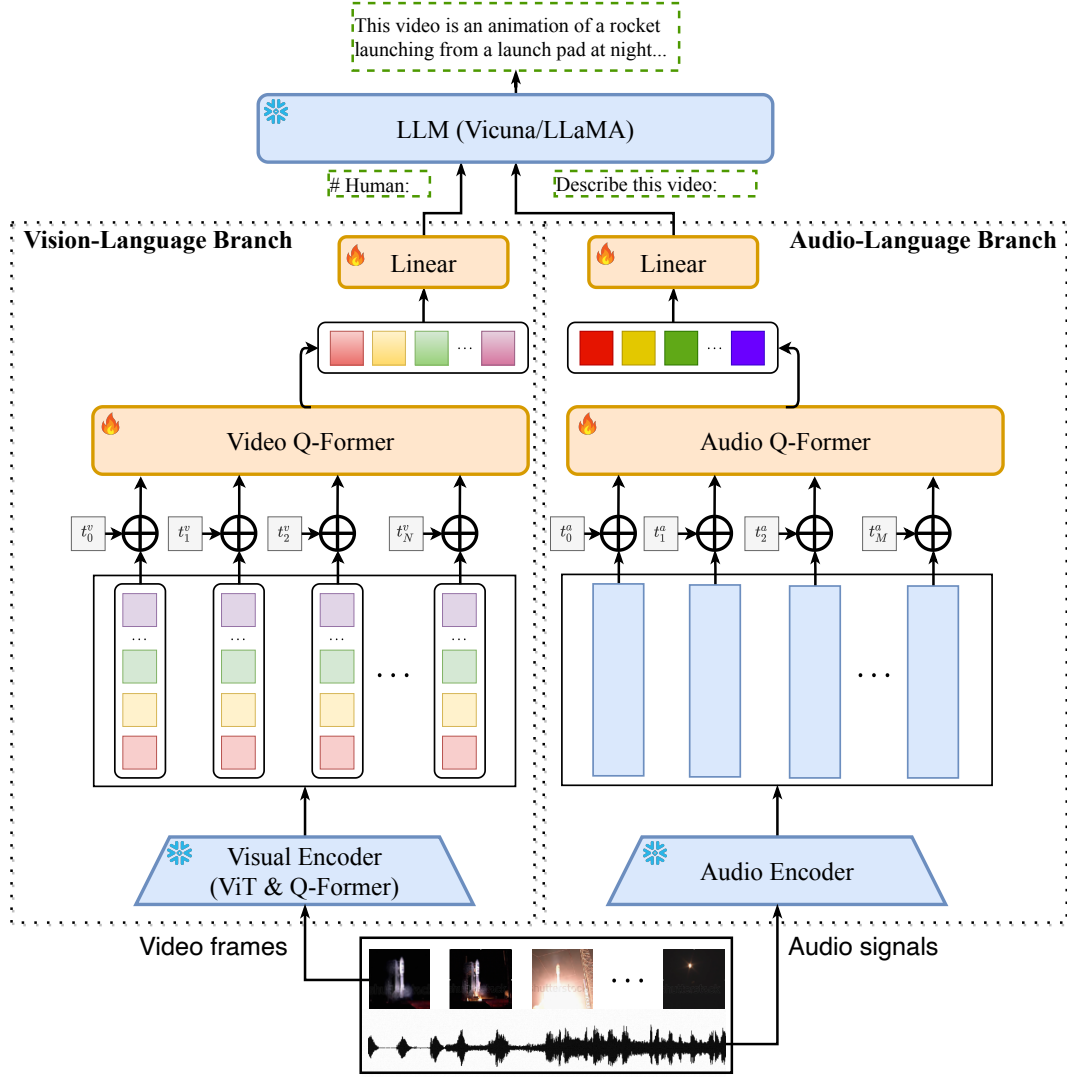


Figure 1: Overall architecture of Video-LLaMA.

## 2.1 Architecture

### 2.1.1 Vision-Language Branch

The Vision-Language Branch is designed for enabling the LLMs to understand visual inputs. As shown in the left part of Figure 1, it is composed of a frozen pre-trained image encoder to extract features from video frames, a position embedding layer to inject temporal information into video frames, a video Q-former to aggregate frame-level representations and a linear layer to project the output video representations into the same dimension as the text embeddings of LLMs. Given one video consists of  $N$  frames, the image encoder will first map each frame/image into  $K_f$  image embedding vectors, yielding video frame representations  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N]$  where  $\mathbf{v}_i \in \mathbb{R}^{K_f \times d_f}$  is the set of  $d_f$ -dimensional image embeddings corresponding to the  $i$ -th frame.

Since the frame representations  $\mathbf{v}_i$  from the frozen image encoder are computed without considering any temporal information, we further apply position embeddings as the indicator of temporal information to the representations from different frames. Then, we feed the position-encoded frame representations to Video Q-former, which shares the same architecture with Query Transformer (Q-Former) in BLIP-2 (Li et al., 2023b), to obtain  $k_V$  video embedding vectors of dimension  $d_v$  as the representation  $\hat{\mathbf{v}} \in \mathbb{R}^{k_V \times d_v}$  of the video.

To adapt the video representations to the input of LLMs, we add a linear layer to transform the video embedding vectors into the video query vectors. The video query vectors are of the same dimension as the text embeddings of LLMs. In the forward pass, they will be concatenated to text embeddings as a *video soft prompt* and guide the frozen LLMs

to generate text conditioned on video content.

As for the implementation of the Vision-Language Branch, we utilize the pre-trained vision component of BLIP-2 (Li et al., 2023b) as the frozen visual encoder, which includes a ViT-G/14 from EVA-CLIP (Fang et al., 2022) and a pre-trained Q-former. The remaining components, including the position embedding layer, Video Q-former, and Linear layer are randomly initialized and optimized to well connect the output of the frozen visual encoder to frozen LLMs.

### 2.1.2 Audio-Language Branch

To deal with the auditory content of the given video, we introduce the Audio-Language Branch. Concretely, it consists of a pre-trained audio encoder to compute features given a short segment of origin audio, a position embedding layer to inject temporal information to audio segments, an audio Q-former to fuse the features of different audio segments, and a linear layer to map the audio representation into the embedding space of LLMs.

In practice, we utilize the pre-trained Imagebind (Girdhar et al., 2023) as the audio encoder. We first uniformly sample  $M$  segments of 2-second short audio clips from the video, then convert each 2-second audio clip into spectrograms using 128 mel-spectrogram bins. After obtaining the spectrogram list of input audio, the audio encoder will map each spectrogram into a dense vector. So the generated audio representation of the given video can be denoted as  $A = [a_1, a_2, \dots, a_M]$ .

Similar to Video Q-Former, the Audio Q-former injects temporal information by adding learnable positional embeddings to audio segments. It then generates fixed-length audio features by computing the interaction across the position-encoded audio segments. Audio Q-Former adopts the same architecture as Q-Former. It projects the variable-length audio representation list  $A$  into a fixed-length sequence  $\hat{\mathbf{A}} \in \mathbb{R}^{K_a \times d_a}$ , where the  $K_a$  is the number of audio embedding vectors and  $d_a$  is the dimension of each vector. Finally, we employ a linear layer to map audio features to the embedding space of the LLM.

## 2.2 Multi-branch Cross-Modal Training

We train the vision-language and audio-language branches separately. In the first stage, large-scale vision-caption datasets are used for training, and in the second stage, high-quality instruction-following datasets were used for fine-tuning. The

image is treated as a one-frame video.

### 2.2.1 Training of Vision-Language Branch

For pre-training vision-language branch, we utilized Webvid-2M (Bain et al., 2021), a large-scale dataset of short videos with textual descriptions sourced from stock footage sites. Moreover, we employed the image caption dataset CC595k, which is sourced from CC3M (Sharma et al., 2018) and filtered by Liu et al. (2023). We adopt a video-to-text generation task during the pre-training stage, i.e., given the representation of a video, prompting the frozen LLM to generate the corresponding text description. We find that a significant portion of textual descriptions are insufficient to reflect the entire content of the videos. Therefore, the visual semantics in the videos are not fully aligned with the textual semantics in the video descriptions. Nevertheless, this stage aimed to utilize a vast amount of data and enable video features to contain as much visual knowledge as possible. We left the abilities of vision-text alignment and instruction-following for the next stage.

After the pre-training stage, the model can generate content about information in the video, but its ability to follow instructions has decreased. Therefore, in the second stage, we fine-tuned the model using high-quality instruction data. We integrated the image-detail-description dataset from MiniGPT-4 (Zhu et al., 2023), the image-instruction dataset from LLaVA (Liu et al., 2023), and the video-instruction dataset from Video-Chat (Li et al., 2023c). After fine-tuning, Video-LLaMA exhibited remarkable abilities in following instructions and comprehending images and videos.

### 2.2.2 Training of Audio-Language Branch

Training the audio-language branch directly using audio-text data is highly challenging due to the rarity of such data. The objective of the learnable parameters in the audio-language branch is to align the output embedding of the frozen audio encoder with the embedding space of LLM. Given the scarcity of audio-text data, we employ a workaround strategy to achieve this objective. ImageBind, which is used as our audio encoder, has a remarkable ability to align different modalities' embeddings to one common space, demonstrating impressive performance on cross-modal retrieval and generation tasks. In light of the scarcity of audio-text data and the abundance of visual-text data, we train the audio-language branch using visual-text

data, following the same data and process as the vision branch. Thanks to the shared embedding space provided by ImageBind, Video-LLaMA exhibits the ability to comprehend audio during inference, even though the audio interface has never been trained on audio data.

### 3 Related Works

**Large Language Models:** Large language models (LLMs) (Black et al., 2022; Scao et al., 2022; OpenAI, 2023; Tsimpoukelli et al., 2021) have demonstrated remarkable language understanding and reasoning abilities, enabling the generation of high-quality natural language text across various domains, including articles, conversations, stories, and poetry. LLMs have already sparked a technological revolution and have been widely applied in different applications. Moreover, a series of open source large models, such as LLaMA (Touvron et al., 2023), BLOOM (Scao et al., 2022) and OPT (Zhang et al., 2022), have greatly promoted technological advancement and made outstanding contributions to the NLP community. Building upon these LLMs, researchers have further extended their capabilities and developed excellent models for various NLP tasks. Examples include Vicuna (Chiang et al., 2023) and Baize (Xu et al., 2023a). Our work is based on these LLMs and provides plug-and-play plugins that empower them with the capability of comprehending both visual and auditory content in videos.

**Multi-modal Large Language Models:** Researchers have been actively exploring the use of LLMs for processing multi-modal inputs (Gao et al., 2023; Li et al., 2023c). Existing approaches can be categorized into two main groups. The first category involves employing LLMs as controllers and utilizing existing multi-modal models as tools. In this approach, when receiving the user’s text instruction, the LLM recognizes the user’s intention and makes decisions about which tools to call. It then generates comprehensive responses by incorporating the results obtained from these off-the-shelf multi-modal models. Examples include ChatGPT (Wu et al., 2023a), HuggingGPT (Shen et al., 2023), and AudioGPT (Huang et al., 2023a). The second category focuses on training fundamental large-scale multi-modal models. The key idea of this line of work is to align the pre-trained foundation models for other modalities to textual LLMs. For instance, Flamingo (Alayrac et al.,

2022a) utilizes a perceiver resampler and a gated cross-attention layer to connect a frozen image encoder and LLM. BLIP2 (Li et al., 2023b) introduces a Q-Former to map learned image queries to the textual embedding space of LLMs. (Liu et al., 2023), mPLUG-owl (Ye et al., 2023) and MiniGPT4 (Zhu et al., 2023) develop instruction-following image-LLMs using image-instruction-following dataset. Video-Chat (Li et al., 2023c) and Video-ChatGPT (Maaz et al., 2023) extend image encoders to video encoders and connect them with LLMs to understand visual content in videos. PandaGPT (Su et al., 2023) utilizes multi-modal encoders from ImageBind, trained exclusively on image-instruction pairs, to enable large models to understand six modalities. Our work falls into the second category, where we train fundamental models to comprehend both the visual and auditory content in videos.

### 4 Examples

In this section, we show some cases to demonstrate Video-LLaMA’s multi-modal instruction-following capability in video/audio/image-grounded conversations. Figure 2 shows some of these examples, while additional cases can be found in Appendix A.

#### (1) Audio-visual integration perception ability.

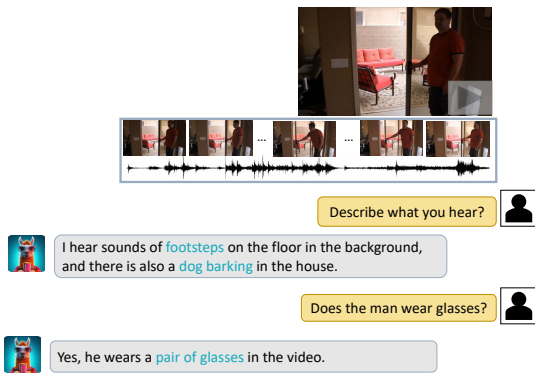
Figure 2(a) and Figure 3 show Video-LLaMA’s unique ability to comprehend auditory and visual information simultaneously. The videos in both cases contain audio. In each conversation, we pose two questions related to visual and auditory content respectively. If the model could only receive one modal, it would be unable to answer both of these questions. However, we can observe that Video-LLaMA accurately responds to both visual and auditory questions in both cases.

#### (2) The ability to capture temporal dynamics in videos.

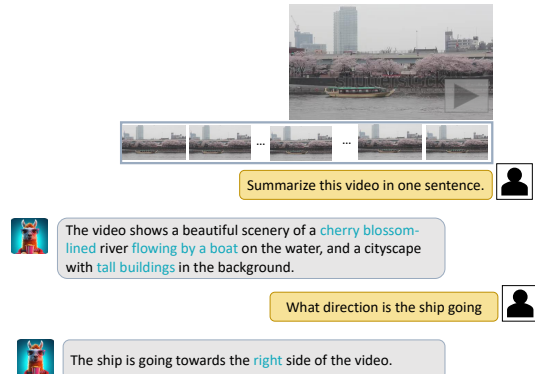
Figure 2(b) and Figure 4 illustrate the capability of Video-LLaMA to identify actions over time. It successfully describes the actions of the girl and the moving direction of the boat.

#### (3) The ability to perceive and understand static images.

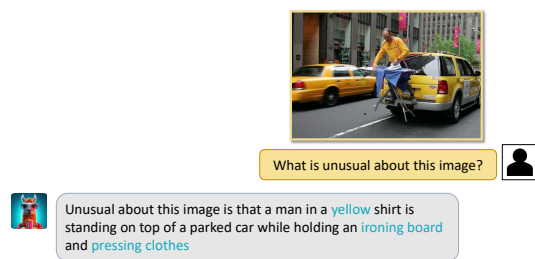
Figure 2(c) and Figure 5 show Video-LLaMA’s ability to perceive and understand pictures. Figure 2(c) demonstrates Video-LLaMA’s ability to understand the concept of "unusual" and specifically describe the unusual scene. In Figure 5, not only does Video-LLaMA accurately describe



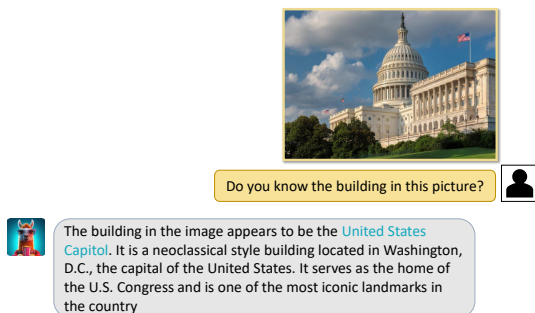
(a) A case that Video-LLaMA answers the questions based on the background sound and visual content of the video



(b) A case showing Video-LLaMA's ability to identify actions over time.



(c) A case demonstrating Video-LLaMA's ability to comprehend static images.



(d) A case demonstrating Video-LLaMA's ability to recognize famous landmarks.

Figure 2: Some examples generated by Video-LLaMA.

the main content, but it also associates it with the friendly interaction between a dog and a human.

**(4) The ability of common-knowledge concept recognition.** Figure 2(d) and Figure 6 demonstrate Video-LLaMA's remarkable capacity for recognizing common-knowledge concepts in visual signals. Video-LLaMA successfully recognizes famous landmarks and characters and can engage in common-sense question-answering.

## 5 Conclusion

In this paper, we present Video-LLaMA, a cutting-edge multi-modal framework that empowers large language models with both audio & video understanding capabilities. Our experiments demonstrated the impressive abilities of Video-LLaMA in audio and video-grounded conversations, highlighting its potential as a promising prototype for audio-visual AI assistants. We have open-sourced the entire training code and various model weights, along with detailed instructions to assist developers in utilizing our code for further development. In addition, we have provided online demo websites and offline demo deployment guides for users to experi-

ence Video-LLaMA's capabilities firsthand. We are committed to constantly maintaining and improving Video-LLaMA, and will continue to contribute to the open-source community.

## 6 Limitations

Although Video-LLaMA has demonstrated impressive abilities in understanding both visual and auditory content in videos, it is still an early-stage prototype and has some limitations, including: (1) Limited perception capacities: Video-LLaMA's performance is hindered by the quality and scale of the current training dataset. We are actively constructing a high-quality audio-video-text alignment dataset to enhance the model's perception capabilities. (2) Limited ability to handle long videos. Long videos (such as movies, and TV shows) contain a large volume of information and impose higher demands on computational resources. This challenge remains a crucial issue that the research community is actively working to address. (3) Hallucination. Video-LLaMA inherits the hallucination problem from the frozen LLMs. We will continue to address these challenges and develop more powerful versions for video understanding.



## References

- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022a. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022b. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- 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 *IEEE International Conference on Computer Vision*.
- Sid 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. *arXiv preprint arXiv:2204.06745*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality.
- 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*.
- Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. 2022. Eva: Exploring the limits of masked visual representation learning at scale. *arXiv preprint arXiv:2211.07636*.
- Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, W. Zhang, Pan Lu, Conghui He, Xiangyu Yue, Hongsheng Li, and Yu Jiao Qiao. 2023. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*.
- Rohit Girdhar, Alaeldin El-Nouby, Zhuang Liu, Manmeet Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. 2023. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190.
- Rongjie Huang, Mingze Li, Dongchao Yang, Jiaotong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 2023a. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*.
- Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, Kriti Aggarwal, Zewen Chi, Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, and Furu Wei. 2023b. Language is not all you need: Aligning perception with language models. *arXiv preprint arXiv:2302.14045*.
- Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. 2023a. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*.
- 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*.
- Kunchang Li, Yanan He, Yi Wang, Yizhuo Li, Wen Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023c. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.
- Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Ming-Hui Qiu, Pengcheng Lu, Tao Wang, and Zhongyu Wei. 2023. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*.
- Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*.
- OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Elie 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*.
- Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual*

- Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565. Association for Computational Linguistics.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.
- Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. 2023. Pandagpt: One model to instruction-follow them all. *arXiv preprint arXiv:2305.16355*.
- Quan Sun, Qiyang Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yuezhe Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. 2023. Generative pretraining in multimodality. *arXiv preprint arXiv:2307.05222*.
- 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*.
- Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212.
- Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*.
- Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.
- Jian Wu, Yashesh Gaur, Zhuo Chen, Long Zhou, Yilun Zhu, Tianrui Wang, Jinyu Li, Shujie Liu, Bo Ren, Linqun Liu, and Yu Wu. 2023b. On decoder-only architecture for speech-to-text and large language model integration. *arXiv preprint arXiv:2307.03917*.
- Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023a. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.
- Haiyang Xu, Qinghao Ye, Mingshi Yan, Yaya Shi, Jiabo Ye, Yuanhong Xu, Chenliang Li, Bin Bi, Quchen Qian, Wei Wang, Guohai Xu, Ji Zhang, Songfang Huang, Feiran Huang, and Jingren Zhou. 2023b. mplug-2: A modularized multi-modal foundation model across text, image and video. *arXiv preprint arXiv:2302.00402*.
- Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yi Zhou, Junyan Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, Chenliang Li, Yuanhong Xu, Hehong Chen, Junfeng Tian, Qiang Qi, Ji Chao Zhang, and Feiyan Huang. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.
- Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*.
- Ao Zhang, Hao Fei, Yuan Yao, Wei Ji, Li Li, Zhiyuan Liu, and Tat-Seng Chua. 2023a. Transfer visual prompt generator across llms. *arXiv preprint arXiv:23045.01278*.
- Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Peng Peng Wang, Yaqian Zhou, and Xipeng Qiu. 2023b. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. *arXiv preprint arXiv:2305.11000*.
- 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*.
- Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

## A Appendix

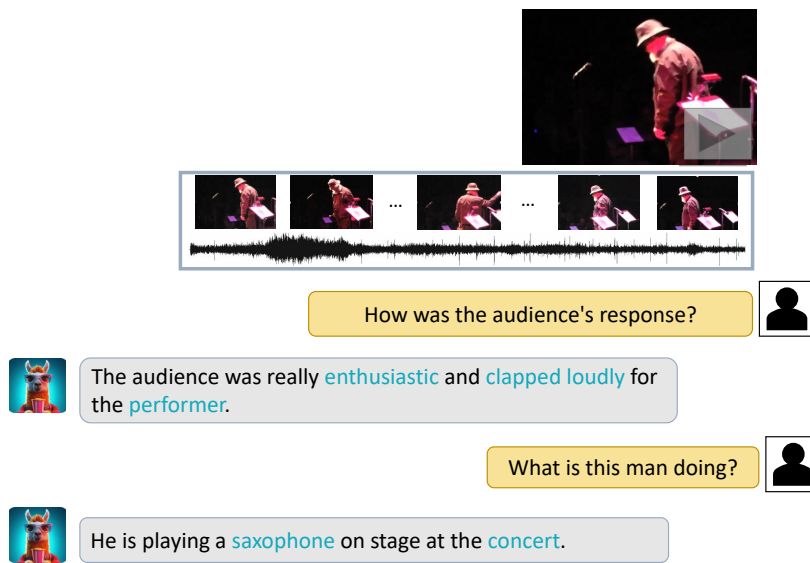


Figure 3: A case showing Video-LLaMA’s ability to identify the sound of applause in a video and infer the positive response from the audience. Additionally, it infers that a man is playing the saxophone on stage based on the visual content.

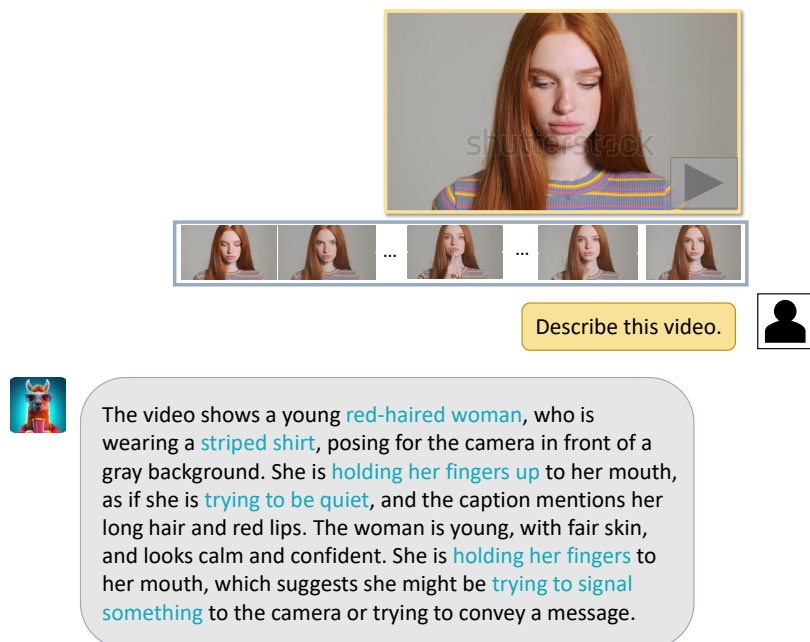
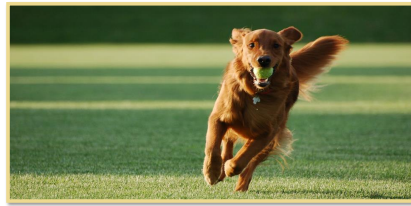


Figure 4: A case where Video-LLaMA provides a detailed description of the visual content in a dynamic video.



Describe the image in details.



The image features a beautiful and playful **golden retriever** running on a bright green lawn. The dog is **running** with a **tennis ball** in its mouth, as it is an active and lively animal that enjoys playing and having fun. **The green grass** in the background creates a vibrant contrast with the golden color of the dog's coat, adding a touch of color and texture to the scene. The dog's fur is well-groomed, shiny, and nicely groomed, highlighting its health and appearance. The brightness of the green lawn and the playful behavior of the dog suggest a joyful atmosphere, inviting viewers to engage in this lovely scenery and the dog's energetic play.

It's worth mentioning that the dog's focus and dedication to playing with the ball also showcase its natural instincts and its hunting abilities. Additionally, this image captures a fun and heartwarming moment of bonding between the dog and its human companion, as the dog's positive attitude towards the ball reflects its enjoyment of interacting with its owner and its environment. Overall, the image conveys an inviting and joyful atmosphere.

Figure 5: A case where Video-LLaMA provides a detailed description of the static image content.

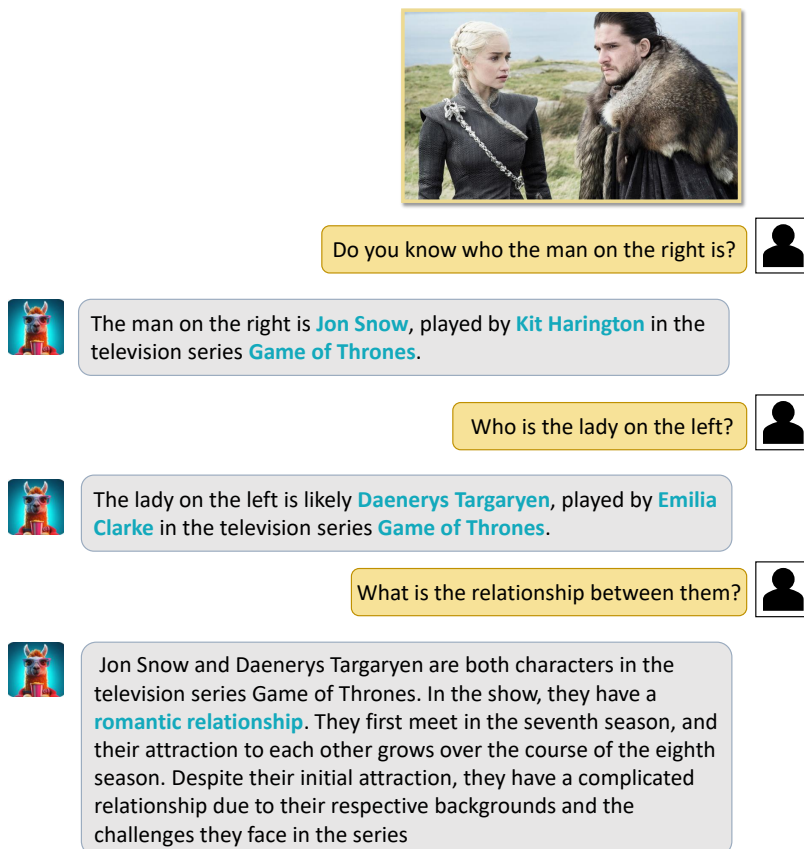


Figure 6: A case showing Video-LLaMA’s ability to recognize renowned characters and participate in video-grounded question answering.

# SUMMHELPER: Collaborative Human-Computer Summarization

Aviv Slobodkin<sup>1\*</sup>, Niv Nachum<sup>1\*</sup>, Shmuel Amar<sup>1</sup>, Ori Shapira<sup>2†</sup>, Ido Dagan<sup>1</sup>

<sup>1</sup>Bar-Ilan University <sup>2</sup>Amazon  
{lovodkin93, niv252, shmulikamar, obspp18}@gmail.com  
dagan@cs.biu.ac.il

## Abstract

Current approaches for text summarization are predominantly automatic, with rather limited space for human intervention and control over the process. In this paper, we introduce SUMMHELPER,<sup>1</sup> a 2-phase summarization assistant designed to foster human-machine collaboration. The initial phase involves content selection, where the system recommends potential content, allowing users to accept, modify, or introduce additional selections. The subsequent phase, content consolidation, involves SUMMHELPER generating a coherent summary from these selections, which users can then refine using visual mappings between the summary and the source text. Small-scale user studies reveal the effectiveness of our application, with participants being especially appreciative of the balance between automated guidance and opportunities for personal input.

## 1 Introduction

Text summarization is the task of generating a condensed version of a given text. Most summarization approaches operate in a fully automated pipeline. While efficient, fully automatic summarization does not flexibly enable human intervention and control during the summarization process, which could potentially tune the process to better accommodate user preferences, as well as rectify inevitable mistakes made by models. Our objective in this paper is to promote such a human-involved approach to summarization, allowing to better tailor the eventual output to real-world user needs, and to synergize the efficiency of the computer with the quality of the human (Hoc, 2000; Pacaux-Lemoine

et al., 2017; Flemisch et al., 2019). The process can conveniently support a range of practical scenarios that require individual preferences, such as editors preparing summaries of articles, students condensing notes, or legal practitioners abridging contracts.

To advance such direction, we present SUMMHELPER, a 2-stage summarization assistant, which decomposes the summarization pipeline into two natural subtasks—content selection followed by summary generation—and facilitates human-machine cooperation in each subtask. On an input document, the process starts with the selection of content for the summary (§3.1). SUMMHELPER suggests possible salient content, efficiently pointing users to central information within the text (see [1] in Figure 1a). Users may accept or reject suggested spans, or highlight any other content to include in the summary ([3] in Figure 1a).

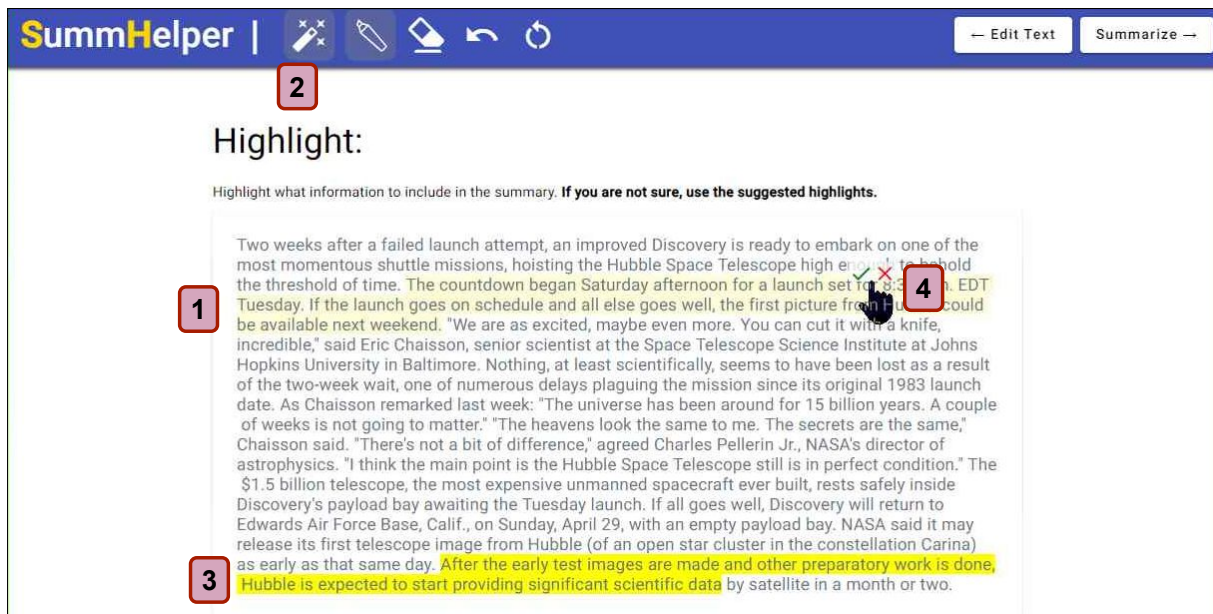
Upon receiving highlighted content within the text, SUMMHELPER subsequently consolidates it and generates a coherent summary (§3.2). This step coincides with the recently introduced Controlled Text Reduction task (CTR; Slobodkin et al., 2022), which produces a coherent fused version of the content of marked spans (“highlights”) in a source document, as interpreted within the context of the full text. Once ready, users can review the generated summary and edit any unsatisfactory content. To facilitate inspection, users are presented with a side-by-side display of the summary and the highlighted input (see Figure 1b), with clearly marked alignments between summary spans and corresponding source text spans ([5] and [6] in Figure 1b). The automatic alignments aid users in navigating through the input text and identifying summary content that may need editing.

To assess SUMMHELPER’s usefulness for generating customized summaries, we conduct two user studies (§4), following common human-computer

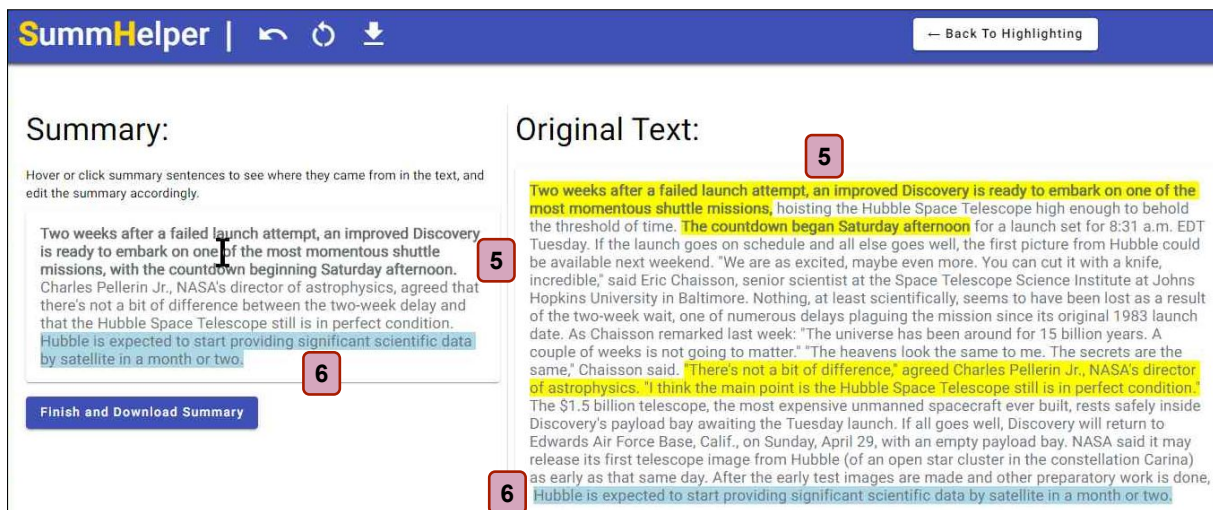
\* Equal contribution.

† Work done in cooperation with Bar-Ilan University (external and not related to the author’s work at Amazon).

<sup>1</sup>System at <https://nlp.biu.ac.il/~sloboda1/SummHelper>, screencast demo at <https://www.youtube.com/watch?v=jKzS9RwuccM> and code is available at <https://github.com/niv252/SummHelper>



(a) Content selection window



(b) Review and editing window

Figure 1: Our SUMMHELPER web application. First, users upload a document and enter the content selection window (1a) to select what information to include in the summary. Users can receive suggestions from the system (pale yellow; [1]), through the magic wand icon [2]. Any part in the text can be highlighted via mouse click-and-drag operations [3]. Users can also accept or reject entire suggested spans via the respective ✓ and ✗ buttons, which appear when hovering over suggestions [4]. When finishing highlighting, a summary is generated, and users proceed to the reviewing window (1b), which shows the generated summary and the source text, with highlights, side-by-side. Here, hovering over a summary sentence emboldens that sentence and its corresponding aligned source text [5]. Additionally, clicking a summary sentence assigns a persistent blue background to the aligning texts [6]. Users can edit the summary freely, with alignments updating automatically.

interaction (HCI) methodologies and applying prominent usability questionnaires. These studies indicate the system’s utility and user-friendly design for a thorough collaborative summarization process. Notably, users valued the tool’s guidance throughout the process, while also appreciating their continuous involvement in refining automatic

decisions.

## 2 Background and Related Work

This section provides a brief overview of related lines of work in summarization. These include strategies offering some level of user control (§2.1),

and modular summarization pipelines that separate the task into distinct subtasks (§2.2).

## 2.1 User Impact on the Summarization Process

Several previous lines of research focused on giving users control of the summary content. In tasks like query-focused (Dang, 2006; Baumel et al., 2018) and aspect-based summarization (Ahuja et al., 2022; Yang et al., 2023), the input text is accompanied by a request around which to focus the output summary. This is a common non-interactive approach for guiding summary content. Other works adapt the summarization process to specific users by learning their preferences. Hu et al. (2012) and Tepper et al. (2018) profile users in order to personalize the summary, via previously discussed aspects in conversations and social connections. Similarly, research on active learning collects summary preferences from users and learns their inclinations toward content and format in order to improve the model’s performance (P.V.S and Meyer, 2017; Zarinbal et al., 2019; Gao et al., 2020). In these works, user influence is mainly confined to attributes in the input or during model adaptation, leaving the summarization process itself fully automatic. In contrast, our approach supports complete user control and intervention in both content selection and the post-generation phase.

Another line of work focuses on designing interactive tools that provide users with certain means of intervention *during* the summarization process. Yan et al. (2011) developed a system supporting iterative selection and removal of source sentences in an extractive system summary until a satisfactory summary is obtained. To aid users in making informed decisions, the system helps users track the context in which summary sentences were mentioned in the source texts. Similarly, P.V.S. et al. (2018) introduced a tool where users can iteratively select concepts in a system summary to remove from the summary or upon which to further elaborate. Xie et al. (2023)’s system allows users to edit system summaries by typing text and receiving automatic completion suggestions. Despite facilitating collaboration with users, these tools start with complete generic system summaries before integrating user feedback. Specifically, they are not well-suited for cases where users wish to include content not present in the initial system summary, or for completely changing its content. In contrast,

our system adapts to user feedback throughout the entire process, allowing users to choose what to include in the summary and assisting them in editing the output to further adjust it to their preferences.

Lastly, interactive exploration systems (Shapira et al., 2022) provide updated summaries for given queries. However, unlike SUMMHELPER, such systems aim to allow learning about a topic, rather than generating a coherent fine-tuned summary.

## 2.2 Modular Summarization

SUMMHELPER is a modular system consisting of separate components, each performing one sub-task, allowing user modifications of that sub-task’s output. Such decomposition has been studied before in the context of fully automated summarization, with several works separating the process into salience detection and generation components (Barzilay and McKeown, 2005; Li et al., 2018; Ernst et al., 2022). These works focused on optimizing each component as part of a fully-automatic summarization process in order to improve the overall performance of the model. In contrast, our work uses this modularity to not only improve overall system output, but to also give more control to the user over each step in the summarization process.

## 3 The SUMMHELPER Application

SUMMHELPER is a web application designed for human-computer cooperation in generating human-controlled summaries, shown in Figure 1. It consists of two stages: (i) computer-assisted content selection via highlighting (§3.1), and (ii) automated summary generation according to the selected content followed by machine-assisted reviewing and editing of the generated summary (§3.2).

### 3.1 Personalized Content Selection

The first step focuses on content selection. The information to incorporate in the summary is manually selected by highlighting it via mouse click-and-drag operations ([3] in Figure 1a). Notably, users can also get suggested content from SUMMHELPER ([1], pale yellow), by clicking the magic wand icon ([2]). Users can accept or reject a full suggestion by clicking the ✓ and ✗ buttons, respectively, which appear when hovering over the suggestion ([4]).

To automatically identify suggested highlights, we deploy the ExtractiveSummarizer model from



the TransformerSum library.<sup>2</sup> The model, a RoBERTa<sub>base</sub> (Liu et al., 2019) trained on the CNN/DailyMail summarization dataset (Hermann et al., 2015), operates as a binary classifier. Its function is to assess the significance of each sentence within the text. As a subsequent operation, the application selects the 30% highest-ranking sentences to suggest to the user. The choice of this model was influenced by its popularity among extractive summarizers, which are all trained to predict salience. Yet, it can be easily replaced with other content selection models to cater to varying needs.

We note that these recommendations are primarily applicable for *generic* summaries. The final content selection decision lies with the users, whose judgment and scrutiny of these suggestions, along with the additional selection of non-suggested content, is instrumental in tailoring the summary to their specific preferences.

### 3.2 Content Consolidation

Once all the desirable content is selected, the next step is to properly consolidate it into a coherent summary. In our setting, SUMMHELPER initially auto-generates such a summary, subsequently providing users with guidance for its review and refinement. For the initial auto-consolidation, we deploy an available Controlled Text Reduction model (Slobodkin et al., 2023), which is a Flan-T5<sub>large</sub> model (Chung et al., 2022), finetuned on the highlights-focused CTR dataset.<sup>3</sup> Upon generation, users are presented with the generated summary and the highlighted input text side-by-side (see Figure 1b). This view facilitates reviewing the summary and editing it when identifying unfavorable outcomes, such as the absence of highlighted content or the inclusion of undesired (non-highlighted or hallucinated) content. To facilitate examination of the summary’s compliance with the highlighted content, the user can hover over summary sentences to embolden both the summary sentence and its corresponding alignment in the source text ([5]). An alignment can be permanently emphasized with a blue background by clicking on a summary sentence, which remains unaffected when hovering over other sentences ([6]). To ensure consistent alignment while the summary is being revised by the user, SUMMHELPER monitors writing pauses and re-calculates alignments when a pause exceeds

<sup>2</sup><https://transformersum.readthedocs.io/en/latest/>

<sup>3</sup>For further details, see Appendix B.

| User      | 1  | 2  | 3  | 4    | 5  | 6    |
|-----------|----|----|----|------|----|------|
| SUS Score | 95 | 95 | 90 | 67.5 | 90 | 82.5 |

Table 1: SUS scores for each user, calculated based on the ten SUS question scores (see Appendix C.1).

one second.

Considering the computational demands of continuous on-the-fly re-alignment, and the alignment feature’s primary goal of pointing users to relevant source text sections, we opted for a lexical-matching approach, which is both fast and sufficient for this goal.<sup>4</sup> Our approach locates the longest common subsequence (LCS) between the lemmas of each input sentence and each summary sentence, followed by several heuristics to filter out irrelevant LCSs (see Appendix A for further details).

## 4 Experiments and Evaluation

We assess SUMMHELPER via two user studies with human subjects, using standard human-computer interaction (HCI) questionnaires. In the first study, we examine the usability of SUMMHELPER for carrying out its purpose, i.e., summarizing an article in a collaborative manner, granting control to the user throughout the process. The second study compares SUMMHELPER to a conventional summarization setup, where a standard auto-generated summary can simply be post-edited without any specialized automated assistance, aiming to assess SUMMHELPER’s comparative utility.

### 4.1 Usability Study

**Setup.** This study aims to gather human feedback regarding the usefulness of SUMMHELPER in performing a collaborative, user-guided, summarization process. Following the discount usability testing principle (Nielsen, 1993), which contends that six evaluators are sufficient for prototype evaluation, we employed six participants for this study. To simulate a plausible real-world scenario, participants were given the persona of an intern journalist who is required to use the application for writing a summary of a news article. All participants performed the task twice, over the same two articles, taken from the DUC 2001 dataset,<sup>5</sup> in random order.

<sup>4</sup>Semantic matching was examined during system development, but was found to have little added value with substantially higher latency.

<sup>5</sup><https://duc.nist.gov>

| System Aspect                 | Score     |
|-------------------------------|-----------|
| Highlights suggestion model   | 3.7 (1.0) |
| Alignments algorithm          | 4.3 (1.0) |
| CTR model                     |           |
| Summary coherence             | 4.2 (0.7) |
| Summary non-redundancy        | 4.6 (0.4) |
| Highlights coverage           | 4.7 (0.4) |
| Highlights adherence          | 4.2 (0.7) |
| Overall satisfaction          | 4.0 (0.7) |
| General                       |           |
| Intuitiveness of highlighting | 4.5 (0.4) |
| Likeliness to recommend       | 4.2 (0.7) |

Table 2: The average and (StD) results of the Usefulness questionnaire on the 12 sessions (2 articles for 6 participants). See Appendix C.1 for the full questions.

To assess SUMMHELPER’s helpfulness in different use cases, one article was relatively long, with ~800 tokens, whereas the other contained ~500 tokens.

During the experiments, we observed the users’ activity and employed a “think aloud” technique (Van Someren et al., 1994) to obtain user remarks. Upon completing the summaries of both articles, participants filled out the standard System Usability Scale (SUS) questionnaire (Brooke, 1996) for subjective usability evaluation, consisting of questions regarding the system’s ease of use, ease of learning, and general flow, with an overall score between 0 and 100.

Additionally, after summarizing each article, participants rated the usefulness of various characteristics of the application on a 1 to 5 scale, including the quality of the different models and algorithms used in the system, the intuitiveness of highlighting and unhighlighting content, and the likeliness of them recommending the system. For more details about the setup, including the full list of the SUS questions and our additional questions, see Appendix C.1.

**Results.** Table 1 presents the SUS scores of each of our 6 participants. With the exception of user number 4,<sup>6</sup> the system received scores exceeding 80, thereby affirming the application’s “excellent” usability (UIUX-Trend, 2021). See Table 4 in the Appendix for itemized scores.

<sup>6</sup>This single participant expressed a strong personal preference for a more abstractive automatic summary, even though this is not necessarily a desired goal on its own in our setting.

This favorable trend is further observed in Table 2, which outlines the average ratings on the system features, across the 12 sessions. Overall, users expressed satisfaction with the application, finding SUMMHELPER’s features helpful and intuitive, including the initial highlight suggestions and the alignment feature. Furthermore, the generated summaries by the CTR model were viewed as highly satisfactory, and there was a discernible interest among several participants to incorporate such an application into their everyday work (e.g., for summarizing legal contracts as well as prescription drug information).

During the study, we observed that the majority of users felt that the suggested highlights were particularly useful when navigating through the *longer* article as opposed to the *shorter* one. Nevertheless, all users expressed satisfaction with the overall summarization process of SUMMHELPER for both articles. They particularly appreciated the two-step procedure encompassing content selection and subsequent review, as it facilitated better text comprehension and instilled greater confidence and control in producing the final output. Two users expressed a desire for an option to create more abstractive summaries that are less verbatim relative to the highlights. Addressing this feedback, by training more abstractive CTR models or performing a post-hoc abstraction of the generated summary, is an interesting future direction we plan to explore. See Appendix C.1 for more feedback and issues raised by participants.

## 4.2 Comparative Usability Test

**Setup.** We compared the use of SUMMHELPER with a setup that simulates a conventional approach when working with summarization systems. In such setup, the input text is first generically summarized with an automatic summarization model. That summary can then be manually post-edited to meet the specific preferences of the user. For the summarization model, we used a BART<sub>large</sub> model (Lewis et al., 2019) trained on the CNN/Daily Mail dataset (Hermann et al., 2015),<sup>7</sup> selected for its noticeable popularity. We adapted SUMMHELPER’s front-end for this process in order to eliminate a potential influence caused by the application’s design. The resulting application comprises two steps: the automatic generation of the generic summary

<sup>7</sup><https://huggingface.co/facebook/bart-large-cnn>

| Dimension             | Score     |
|-----------------------|-----------|
| Usefulness            | 4.3 (0.5) |
| Ease of Use           | 3.6 (0.6) |
| Ease of Learning      | 3.1 (0.3) |
| Satisfaction          | 4.1 (0.6) |
| Summarization Process | 4.7 (0.5) |

Table 3: The average (StD) results of the five dimensions in the USE questionnaire. A score of 1 represents a preference for ONLYSUMM and 5 prefers SUMMHELPER.

and the review step for manual editing. During reviewing, users are presented with the input text and the generated summary side-by-side, allowing them to make adaptations to the summary (without the alignment feature). We refer to this adapted application as ONLYSUMM.

For this experiment, we asked 6 new participants to follow the task described in §4.1, which involved summarizing a news article, taking the perspective of an intern journalist, once with SUMMHELPER on one article, and once with ONLYSUMM on another article (with different orders of articles and applications). Upon completion of both sessions, participants filled out a questionnaire, adapted from the standard USE Questionnaire (Lund, 2001). In the questionnaire, 32 statements are rated on a scale of 1 (ONLYSUMM is preferred) to 5 (SUMMHELPER is preferred). The original 30 USE statements represent 4 dimensions: Usefulness, Ease of Use, Ease of Learning, and Satisfaction (see Appendix C.2 for the full list of statements). We also added 2 statements to rank users’ experience with the key aspects of our summarization process (represented as the fifth dimension in Table 3). These additional statements were: “I found it easy to control what information to include in the final summary” and “I found it easy to make sure the final summary had all the information I wanted”. More details elaborating on the study are available in Appendix C.2.

**Results.** Table 3 presents the scores for each dimension examined, averaged over the corresponding statements and the six participants. Interestingly, despite SUMMHELPER consisting of more features and steps than ONLYSUMM, participants did not find it more challenging to learn. Moreover, they reported that SUMMHELPER was somewhat more user-friendly. SUMMHELPER was strongly favored over ONLYSUMM in terms of Usefulness, Satisfaction, and, notably, the Sum-

marization Process, underscoring the practicality of SUMMHELPER for preparing customized summaries.

Importantly, we observed that users tended to be very meticulous when summarizing with SUMMHELPER, exhibiting a higher inclination to carefully inspect the text and critically evaluate the inclusion of each piece of information. Indeed, even with the suggested highlights, users cautiously appraised each suggestion and more often selected only sub-segments of it. In contrast, we found that when summarizing with ONLYSUMM, participants typically skimmed the input text and accepted the generated summaries with minimal adjustments. Therefore, although using SUMMHELPER generally took longer to summarize (11.1 minutes on average, compared to 7.0 minutes with ONLYSUMM), it led to a more thorough summarization process. This is corroborated by the Usefulness, Satisfaction, and Summarization Process scores in Table 3, and participants’ feedback, which consistently indicated higher confidence and satisfaction with their completed work when using SUMMHELPER.

## 5 Conclusion

In this paper, we presented SUMMHELPER, a novel summarization assistant, which collaborates with users across two steps: content selection and content consolidation. The system facilitates user intervention and supervision along the summarization process, in order to achieve the most suitable output tailored to specific needs. Preliminary user studies illustrate SUMMHELPER’s potential for a thorough and collaborative summarization process, with users expressing satisfaction with the process, as well as the final output.

Future work may include investigating more effective semantic strategies to locate summary-source alignments with acceptable latency. Additionally, in light of some user feedback, another interesting extension includes developing more abstractive consolidation and fusion models, which would offer control over the level of abstractness in the outputs. Lastly, exploring strategies to scale SUMMHELPER to a multi-document setting presents another promising avenue for future investigation.

## Limitations

This demo focuses on the single-document setting. Future work should expand the application’s capa-

bilities to the multi-document setting, both in terms of the backend models and in terms of accessibility and intuitiveness of the application’s frontend design. Additionally, our tool currently helps users in the reviewing step solely with the alignment functionality. Future work should add additional assistance during this step in the form of suggested improvements to selected unsatisfactory content in the summary, in addition to the alignment feature.

## Ethics Statement

We conducted the usability (§4.1) and comparative usability (§4.2) studies in person. Participants volunteered to take part in the study, taking about 40 minutes for the former experiment, or 35 minutes for the latter. A consent form was signed by participants prior to each session, which stressed the fact that the user study was voluntary and that they were encouraged to withdraw if they felt any discomfort. In addition, the form ensured that the participant is at least 18 years of age, and assured that personal details remain anonymous.

The source texts (news articles) used in the user studies were acquired according to the required NIST guidelines (<https://duc.nist.gov>).

## Acknowledgements

This work was supported by the Israel Science Foundation (grant no. 2827/21), and a grant from the Israel Ministry of Science and Technology. We would also like to thank Hadar Ronen for her guidance in planning the user studies.

## References

- Ojas Ahuja, Jiacheng Xu, Akshay Gupta, Kevin Horecka, and Greg Durrett. 2022. [ASPECTNEWS: Aspect-Oriented Summarization of News Documents](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6494–6506, Dublin, Ireland. Association for Computational Linguistics.
- Regina Barzilay and Kathleen R. McKeown. 2005. [Sentence Fusion for Multidocument News Summarization](#). *Computational Linguistics*, 31(3):297–328.
- Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. [Query Focused Abstractive Summarization: Incorporating Query Relevance, Multi-Document Coverage, and Summary Length Constraints into seq2seq Models](#).
- John Brooke. 1996. [SUS: A Quick and Dirty Usability Scale](#). *Usability evaluation in industry*, 189(3):189–194.
- 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 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](#).
- Hoa Trang Dang. 2006. [DUC 2005: Evaluation of Question-Focused Summarization Systems](#). In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, Sydney, Australia. Association for Computational Linguistics.
- Ori Ernst, Avi Caciularu, Ori Shapira, Ramakanth Pasunuru, Mohit Bansal, Jacob Goldberger, and Ido Dagan. 2022. [Proposition-Level Clustering for Multi-Document Summarization](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1765–1779, Seattle, United States. Association for Computational Linguistics.
- Frank Flemisch, David A Abbink, Makoto Itoh, M-P Pacaux-Lemoine, and Gina Wessel. 2019. [Joining the blunt and the pointy end of the spear: towards a common framework of joint action, human-machine cooperation, cooperative guidance and control, shared, traded and supervisory control](#). *Cognition, Technology & Work*, 21:555–568.
- Yang Gao, Christian M. Meyer, and Iryna Gurevych. 2020. [Preference-Based Interactive Multi-Document Summarisation](#). *Information Retrieval*, 23(6):555–585.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching Machines to Read and Comprehend](#).
- Jean-Michel Hoc. 2000. [From human-machine interaction to human-machine cooperation](#). *Ergonomics*, 43(7):833–843.
- Po Hu, Donghong Ji, Chong Teng, and Yujing Guo. 2012. [Context-Enhanced Personalized Social Summarization](#). In *Proceedings of COLING 2012*, pages 1223–1238, Mumbai, India. The COLING 2012 Organizing Committee.
- 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](#). *CoRR*, abs/1910.13461.

- Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. [Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#). *CoRR*, abs/1907.11692.
- Arnold M Lund. 2001. [Measuring Usability with the USE Questionnaire](#). *Usability interface*, 8(2):3–6.
- Jakob Nielsen. 1993. [Usability Engineering](#).
- Marie-Pierre Pacaux-Lemoine, Damien Trentesaux, Gabriel Zambrano Rey, and Patrick Millot. 2017. [Designing intelligent manufacturing systems through Human-Machine Cooperation principles: A human-centered approach](#). *Computers & Industrial Engineering*, 111:581–595.
- Avinesh P.V.S., Benjamin Hättasch, Orkan Özyurt, Carsten Binnig, and Christian M. Meyer. 2018. [Sherlock: A System for Interactive Summarization of Large Text Collections](#). *Proc. VLDB Endow.*, 11(12):1902–1905.
- Avinesh P.V.S and Christian M. Meyer. 2017. [Joint Optimization of User-desired Content in Multi-document Summaries by Learning from User Feedback](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1353–1363, Vancouver, Canada. Association for Computational Linguistics.
- Ori Shapira, Ramakanth Pasunuru, Mohit Bansal, Ido Dagan, and Yael Amsterdamer. 2022. [Interactive Query-Assisted Summarization via Deep Reinforcement Learning](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2551–2568, Seattle, United States. Association for Computational Linguistics.
- Aviv Slobodkin, Avi Caciularu, Eran Hirsch, and Ido Dagan. 2023. [Dont add, dont miss: Effective content preserving generation from pre-selected text spans](#).
- Aviv Slobodkin, Paul Roit, Eran Hirsch, Ori Ernst, and Ido Dagan. 2022. [Controlled Text Reduction](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5699–5715, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Naama Tepper, Anat Hashavit, Maya Barnea, Inbal Ronen, and Lior Leiba. 2018. [Collabot: Personalized Group Chat Summarization](#). In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM ’18, page 771–774, New York, NY, USA. Association for Computing Machinery.
- UIUX-Trend. 2021. [Measuring and Interpreting System Usability Scale - UIUX Trend](#). <https://uiuxtrend.com/measuring-system-usability-scale-sus/>. Accessed: 2023-08-01.
- Maarten Van Someren, Yvonne F Barnard, and J Sandberg. 1994. [The Think Aloud Method: A Practical Approach to Modelling Cognitive Processes](#). *London: AcademicPress*, 11:29–41.
- Yujia Xie, Xun Wang, Si-Qing Chen, Wayne Xiong, and Pengcheng He. 2023. [Interactive Editing for Text Summarization](#).
- Rui Yan, Jian-Yun Nie, and Xiaoming Li. 2011. [Summarize What You Are Interested In: An Optimization Framework for Interactive Personalized Summarization](#). In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1342–1351, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Xianjun Yang, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Xiaoman Pan, Linda Petzold, and Dong Yu. 2023. [OASum: Large-Scale Open Domain Aspect-based Summarization](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4381–4401, Toronto, Canada. Association for Computational Linguistics.
- Marzieh Zarinbal, Azadeh Mohebi, Hesamoddin Mosalli, Razieh Haratinik, Zahra Jabalameli, and Farnoush Bayatmakou. 2019. [A New Social Robot for Interactive Query-Based Summarization: Scientific Document Summarization](#). In *Interactive Collaborative Robotics*, pages 330–340, Cham. Springer International Publishing.

## A Alignment Algorithm

In the reviewing phase, the system aids users in comparing the highlighted input text to the generated output summary, in order to spot any potential disapprovals in the summary. This is achieved by automatically identifying text from the input that aligns with each sentence in the summary, and clearly marking it (§3.2) for the user. To find these alignments, the system first performs sentence tokenization on the input source text and the generated summary. For each pair of summary and input sentences, it then calculates the longest common subsequence (LCS) of their lemmas.

To filter out insignificant alignments, LCSs containing less than three content tokens (neither stop words nor punctuation), denoted *short LCSs*, are disregarded. For instance, as demonstrated in [Figure 2](#), the LCS “John eat today” between the first

sentences of the summary and input consists of three content words and is thus preserved. In contrast, the LCS “Mr. Smith” between the first summary sentence and the second input sentence, having only two content words, is discarded. For alignment *within highlights*, a short LCS is still retained if it covers at least 25% of the highlighted span’s content lemmas. For instance, even though the LCS “he call me” of the last sentences of the summary and input in Figure 2 contains only one content lemma (“call”), it covers 100% of the highlight’s content lemmas and is thus retained.

Finally, the alignment algorithm also addresses cases where the CTR model reorders content within input sentences. An LCS procedure is not well-suited for such situations. To this end, the algorithm iteratively calculates four LCSs for each pair of summary and input sentences. After each iteration, the part of the summary sentence contributing to the LCS is omitted, enabling shorter LCSs to be identified. For example, after identifying the LCS of the first sentences of the input and summary in Figure 2 (“John eat today”), the algorithm generates a variant of the summary sentence by excluding the LCS, resulting in “Mr. Smith said early”. It then identifies the LCS “Mr. Smith” between this variant and the first input sentence, which is preserved as it covers 50% of the second highlighted span’s content lemmas (“Mr.”, “Smith”, “tell”, “mother”).

## B CTR Model

Controlled Text Reduction (CTR; Slobodkin et al., 2022), is a recently introduced task, which takes as input a text with pre-selected marked spans (“highlights”) and expects a coherent version of the text, covering exactly the content of these highlights. It handles coherence issues relating to discourse and coreference. This task conforms with our summary generation process, and we hence employ an available Controlled Text Reduction model.<sup>8</sup> This model is a Flan-T5<sub>large</sub> model (Chung et al., 2022), finetuned on the highlights-focused CTR dataset. Following Slobodkin et al. (2022), highlights are incorporated into the input text with special markups, `<extra_id_1>` and `<extra_id_2>`, marking the beginning and end of each highlighted span, respectively. In our configuration, we set the maximum input length to 4096 and the maximum

<sup>8</sup>[https://github.com/lovodkin93/CTR\\_instruction\\_finetuning](https://github.com/lovodkin93/CTR_instruction_finetuning)

### Input text:

...

**John** has already **eaten today**, **Mr. Smith** told his mother.

Mr. Smith didn’t recognize him.

**He** immediately **called me**.

...

### Summary:

...

**Mr. Smith** said **John ate** early **today**.

**He** then **called me**.

...

Figure 2: An example of the alignment algorithm for an extract of the highlighted input text and that of the respective summary. The first lemma-based LCS between the first sentences of the summary and input is “John eat today” (bold and red), which has  $\geq 3$  content words (John, eat, today) and is thus retained. The second LCS, “Mr. Smith”, contains  $\geq 25\%$  of the second highlighted span’s (“Mr.”, “Smith”, “tell”, “mother”) content words, and is also retained. On the other hand, the LCS “Mr. Smith” between the first summary sentence and the second input sentence, having only two content words and lacking overlap with any highlighted span, is filtered out. For the second summary sentence and the third input sentence, the only LCS, “He called me” (bold and green), comprises a single content word (“called”) which covers 100% of the third highlight’s content words and is thereby retained.

target length to 400. A greedy decoding strategy was used in order to optimize the decoding speed. Other parameters are kept consistent with the pre-defined generation parameters of the model.

## C Experimental Details

In this work, we performed a usability study and a system comparison experiment (§4) to assess the utility of our application.

### C.1 System Usability Tests

For the usability study, six participants were gathered based on previous acquaintance. These participants varied in their age (28-33), gender, and occupation. Each session took approximately 40 minutes. A participant started by filling out an experiment participation consent form. Next, the different elements of the application were explained and demonstrated to the participant. Then, the participant was asked to experiment with the appli-

As an intern reporter, your assignment is to study two articles written by a senior journalist, and write a summary for each article, suitable for sharing on social media platforms. This task forms a critical part of your internship evaluation, hence meticulous attention to detail is mandatory. You're granted access to an application that can assist you in accomplishing this task. However, it's crucial that the final summary remains a testament to your individual effort and understanding of the articles.

Figure 3: The instructions given to the user study participants.

cation on an example article, to reduce the learning curve of using the system for the first time. Once this onboarding stage was over, the experimentee was presented with the assignment (see Figure 3). The participants conducted the experiments on two articles, one with ~800 tokens and another with ~500 tokens, in a random order.

**SUS questionnaire.** The System Usability Scale (SUS) questionnaire (Brooke, 1996) was filled out once by each participant after completing both article summaries, with the following 10 questions being rated on a scale from 1 (“strongly disagree”) to 5 (“strongly agree”):

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

| SUS Question   | Average Score |
|--|---------------|
| I think that I would like to use this system frequently.                                   | 4.17 (0.98)   |
| I found the system unnecessarily complex.  | 1.83 (0.98)   |
| I thought the system was easy to use.  | 4.33 (0.52)   |
| I think that I would need the support of a technical person to be able to use this system. | 1.33 (0.82)   |
| I found the various functions in this system were well integrated.                         | 4.17 (0.41)   |
| I thought there was too much inconsistency in this system.                                 | 1.17 (0.41)   |
| I would imagine that most people would learn to use this system very quickly.              | 4.67 (0.52)   |
| I found the system very cumbersome to use.   | 1.33 (0.52)   |
| I felt very confident using the system.  | 4.50 (0.84)   |
| I needed to learn a lot of things before I could get going with this system.               | 1.50 (0.84)   |

Table 4: The average (StD) score of the ten SUS questions asked after the usability study, on a scale of 1 to 5.

The SUS scores (Table 1) were calculated using the procedure by Brooke (1996), as follows. Initially, the score contributions from each item were summed up, with each item’s contribution ranging in a 0 to 4 scale. For the odd-numbered items (1,3,5,7, and 9), the score contribution was determined as the scale position minus 1. Conversely, for the even-numbered items (2,4,6,8, and 10), the contribution was calculated as 5 minus the scale position. This sum was then multiplied by 2.5 to compute the overall SUS value for each user, with scores having a range of 0 to 100. We also calculated the average (StD) score for each question, as delineated in Table 4.

**Usefulness questionnaire.** After summarizing each of the two articles, the users filled out a usefulness questionnaire (see results in Table 2), where they were asked to rate the following 9 questions on a scale of 1 (“strongly disagree”) to 5 (“strongly agree”), addressing the different components in our system:

1. For the requirements of the given task, the initial highlights were very helpful.
2. The alignments were helpful in assessing the content of the final summary.
3. It was intuitive to highlight and unhighlight information.
4. I would recommend this app for another intern journalist in my company.

Overall, the summary output by the system was:

5. Coherent
6. Non-Redundant
7. Highlights were covered fully
8. Did not cover unhighlighted content
9. To my satisfaction

**Comments raised by participants.** During the sessions, we collected comments and ideas for improvements raised by the participants. All the participants were very impressed with the summaries generated by the CTR model. Additionally, several users expressed their satisfaction with the modular process, stating that their continuous involvement was crucial for achieving the optimal summary. Users especially appreciated the side-by-side presentation of the highlighted input text and the summary, combined with the alignment feature, which helped them to both stay connected to the source text and optimize their navigation through it. For improvements, one suggestion was to enable generation of more abstractive summaries, that do not align as much with the highlights' phrasing. Additional suggestions included making a different icon for exiting erase mode and entering highlight mode in the content selection window,<sup>9</sup> enabling a dynamic number of suggestions proportionate to the text's length,<sup>10</sup> and enabling the option to go back to the beginning of the process by clicking the application's name in the toolbar.

## C.2 System Comparison Experiment

For the comparative experiment, we gathered 6 new participants, also based on previous acquaintance. These participants varied in their age (24-35), gender, and occupation. Each session took approximately 35 minutes, which started with a participant filling out the same participation form as in the system usability tests (see Appendix C.1). Similarly to the usability test setting, prior to the actual experiment, the different elements of each of the two applications were explained and demonstrated to the participant, and they were asked to experiment with the system on an article. Once the participant

<sup>9</sup>In the first system version, there was only an icon to enter erasing mode, and in order to exit the erasing mode and enter highlighting mode, users needed to click this icon again.

<sup>10</sup>In the first system version, there were always 3 suggestions.

felt confident with their understanding of each application, they were presented with the assignment in Figure 3 and asked to complete it on the same 2 articles as in the system usability tests, once with SUMMHELPER and once with ONLYSUMM (in different orders and different article-model pairings).

**Questionnaire.** After completing both articles, the participant answered a comparative usability questionnaire, adapted from the standard USE Questionnaire (Lund, 2001), as mentioned in §4.2. The original questionnaire consists of 30 statements, divided into 4 dimensions: Usefulness, Ease of Use, Ease of Learning, and Satisfaction. These questions are:

- Usefulness
  1. It helps me be more effective.
  2. It helps me be more productive.
  3. It is useful.
  4. It gives me more control over output.
  5. It makes it easier to achieve the desired output.
  6. It saves me time when I use it.
  7. It meets my needs in addressing the task.
  8. It does everything I would expect it to do.
- Ease of Use
  9. It is easy to use.
  10. It is simple to use.
  11. It is user-friendly.
  12. It requires the fewest steps possible to accomplish the task.
  13. It is flexible.
  14. Using it is effortless.
  15. I can use it without written instructions.
  16. I don't notice any inconsistencies as I use it.
  17. Both occasional and regular users would like it.
  18. I can recover from mistakes quickly and easily.
  19. I can use it successfully every time.
- Ease of Learning
  20. I learned to use it quickly.
  21. I easily remember how to use it.
  22. It is easy to learn to use it.



23. I quickly became skillful with it.

- Satisfaction

24. I am satisfied with it.

25. I would recommend it to a friend.

26. It is fun to use.

27. It works the way I want it to work.

28. It is wonderful.

29. I feel I need to have it.

30. It is pleasant to use.

For each statement, participants were asked to rate it on a scale from 1 (preferred ONLYSUMM) to 5 (preferred SUMMHELPER). In addition to those statements, we added two more statements, in order to rate the participants' experience with the key aspects of the Summarization Process:

31. I found it easy to control what information to include in the final summary.

32. I found it easy to make sure the final summary had all the information I wanted.

**Observations and general feedback.** Overall, all participants favored SUMMHELPER over ONLYSUMM. They especially appreciated the alignment feature, with one participant who started with SUMMHELPER, and expressed frustration with the absence of the alignment feature in ONLYSUMM. Additionally, we observed that all 6 users were meticulous when working with SUMMHELPER, and appraised each suggestion very carefully, as well as non-suggested content. Alternatively, when working with ONLYSUMM, 4 out of the 6 participants simply skimmed the article and were quick to accept the generated summary with minimal adjustments. This shows SUMMHELPER's potential to foster a more thorough and productive summarization process.

# ModelScope-Agent: Building Your Customizable Agent System with Open-source Large Language Models

Chenliang Li, Hehong Chen, Ming Yan\*, Weizhou Shen, Haiyang Xu, Zhikai Wu  
Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, Hongzhu Shi

Ji Zhang, Fei Huang, Jingren Zhou

DAMO Academy, Alibaba Group, China

{lcl193798, hehong.chh, ym119608, shenweizhou.swz, shuofeng.xhy, wuzhikai.wzk, zhangzhicheng.zzc,  
wenmeng.zwm, yingda.chen, chengchen.cc, hongzhu.shz, zj122146, f.huang, jingren.zhou}@alibaba-inc.com

## Abstract

Large language models (LLMs) have recently demonstrated remarkable capabilities to comprehend human intentions, engage in reasoning, and design planning-like behavior. To further unleash the power of LLMs to accomplish complex tasks, there is a growing trend to build agent frameworks that equips LLMs, such as ChatGPT, with tool-use abilities to connect with massive external APIs.

In this work, we introduce ModelScope-Agent, a general and customizable agent framework for real-world applications, based on open-source LLMs as controllers. It provides a user-friendly system library, with a customizable engine design to support model training on multiple open-source LLMs, while also enabling seamless integration with both model APIs and common APIs in a unified way. To equip the LLMs with tool-use abilities, a comprehensive framework has been proposed spanning tool-use data collection, tool retrieval, tool registration, memory control, customized model training, and evaluation for practical real-world applications. Finally, we showcase ModelScopeGPT<sup>1</sup>, a real-world intelligent assistant of ModelScope Community based on the ModelScope-Agent framework, which is able to connect open-source LLMs with more than 1000 public AI models and localized community knowledge in ModelScope. The ModelScope-Agent online demo<sup>2</sup>, library<sup>3</sup> are now publicly available.

## 1 Introduction

Large language models (OpenAI, 2022, 2023; Touvron et al., 2023; Chowdhery et al., 2022) have gradually become common AI assistants

that demonstrate great potential in comprehending human intentions, performing complex reasoning tasks, and enabling content creation. Despite the rapid advancements of open-source LLMs, e.g., LLaMA (Touvron et al., 2023) and ChatGLM (THUDM, 2023), they still remain limited in performing complex tasks, such as following user instructions to use external tools and capture up-to-date information.

To further unleash the power of LLMs for real-world practical applications, a rising trend of current research (Schick et al., 2023; Shen et al., 2023; Yang et al., 2023; Qin et al., 2023; Patil et al., 2023) begins to enable LLMs with tool-use abilities towards building an AI Agent. These include HuggingGPT (Shen et al., 2023), Visual-ChatGPT (Wu et al., 2023) and Gorilla (Patil et al., 2023) for connecting with HuggingFace models, ToolAlpaca (Tang et al., 2023) and ToolLLaMA (Qin et al., 2023) for using massive common APIs such as weather forecast and search engine. These methods either directly rely on closed-source counterparts like ChatGPT or focus on certain types of API tools. Recently, there have also been public releases of AI agents, such as Auto-GPT<sup>4</sup>, LangChain<sup>5</sup> and Transformers Agent (Huggingface, 2023), which enable LLMs, such as ChatGPT or GPT-4, to use tools and solve complex AI tasks. However, these agents are mainly built with closed-source LLMs and how to build a customizable agent system with open-source LLMs remains largely unexplored.

In this work, we present ModelScope-Agent, a general and customizable agent system for real-world applications, based on open-source LLMs as controllers. ModelScope<sup>6</sup> is a public ML community, that seeks to bring together the most advanced machine learning models from the AI community, and streamlines the process of leveraging

\*Corresponding author: <ym119608@alibaba-inc.com>

<sup>1</sup><https://modelscope.cn/studios/damo/ModelscopeGPT>

<sup>2</sup><https://modelscope.cn/studios/lcl193798/Modelscope-Agent>

<sup>3</sup><https://github.com/modelscope/modelscope-agent>

<sup>4</sup><https://github.com/Significant-Gravitas/Auto-GPT>

<sup>5</sup><https://github.com/langchain-ai/langchain>

<sup>6</sup><https://modelscope.cn/models>

AI models in real-world applications. ModelScope-Agent provides a flexible and user-friendly system library, with a customizable engine design to support model training on multiple open-source LLMs, while also enabling seamless integration with both model APIs and common APIs in a unified way. It features an LLM-centric system design, which includes open-source LLMs as core controller, and further interact with a tool-use module and a memory module to accomplish complex tasks. At the core of ModelScope-Agent, the library supports flexible selection and training on various open-source LLMs, such as LLaMA (Touvron et al., 2023), ChatGLM (THUDM, 2023), ChatPLUG (Tian et al., 2023) and other customized LLMs in ModelScope. For tool use, ModelScope-Agent provides a default tool library, which supports diverse AI model APIs across NLP, CV, Audio and Multi-model fields, as well as massive common APIs such as search engine. It also supports registering new self-defined API plugins and automatic API retrieval from the large tool library. It is easy for users to customize their most appropriate LLMs, local API tools and functions to develop real-world applications. Moreover, a memory module is also introduced to better store and manage the system message, user history, in-context examples, tool message and localized knowledge.

To enable the open-source LLMs to better control the whole agent system, we further propose a comprehensive framework of tool-use data collection, customized model training, evaluation and deployment. Notably, we release a comprehensive tool-enhanced dataset MSAgent-Bench, which consists of 598k dialogues with various API categories, multi-turn API calls, API-Oriented QA, and API-Agnostic instructions in both English and Chinese. A simple training strategy of Weighted LM, that enhances the training of generation of API name and parameters, is used to better ensure the correctness of API calls. Besides, an evaluation framework is also supported in our library to examine the tool-use abilities of the trained models in different aspects. Furthermore, we applied ModelScope-Agent in a real-world application of ModelScope Community namely ModelScopeGPT, which is able to connect open-source LLMs with more than 1000 public AI models and access localized community knowledge in ModelScope for community QA.

To summarize, ModelScope-Agent is a general

and customizable agent system designed for developers to harness the power of open-source LLMs. The library targets the following goals:

- Agent based on Open-Source LLMs: the controller of ModelScope-Agent can be flexibly selected from open-source LLMs that are optimized through our agent training framework.
- Support and Customization of Diverse Tools: Dozens of diverse model APIs and common APIs are given by default. The library supports registering new self-defined APIs and automatic API retrieval from the toolset.
- Customizable of Applications: ModelScope-Agent can be flexibly applied in various industry applications. The agent and training framework are documented describing its usage, construction and optimization.

ModelScope-Agent is in continual development by the engineers at ModelScope and is released under an Apache 2.0 license. Full documentation is available through the project website.

## 2 The ModelScope Agent

ModelScope-Agent is designed to facilitate developers in building customizable agent systems based on open-source LLMs. The overall system architecture is shown in Figure 1. It includes open-source LLMs as controller, a tool-use module and a memory module to interact with. Given human instruction, the Agent, which adopts the selected LLM as the controller, will automatically plan tasks, selectively use tools, leverage knowledge in memory, and finally provide helpful responses to users.

### 2.1 LLMs as Brain

LLMs serve as the brain of the agent, responsible for planning and decomposing user requests, selectively calling tools, performing retrieval, and integrating all the information from previous steps to generate the final response. In order to make it easier for users to customize the agent with their own LLMs, we have added support for various open-source LLMs by default, such as LLaMA, ChatGLM and ChatPLUG, which have been optimized through our tool learning pipeline. The details of the training strategy and tool-use datasets can be referred to in Section 3. ModelScope-Agent has integrated the LLM inference pipeline of the ModelScope community, and replacing LLMs

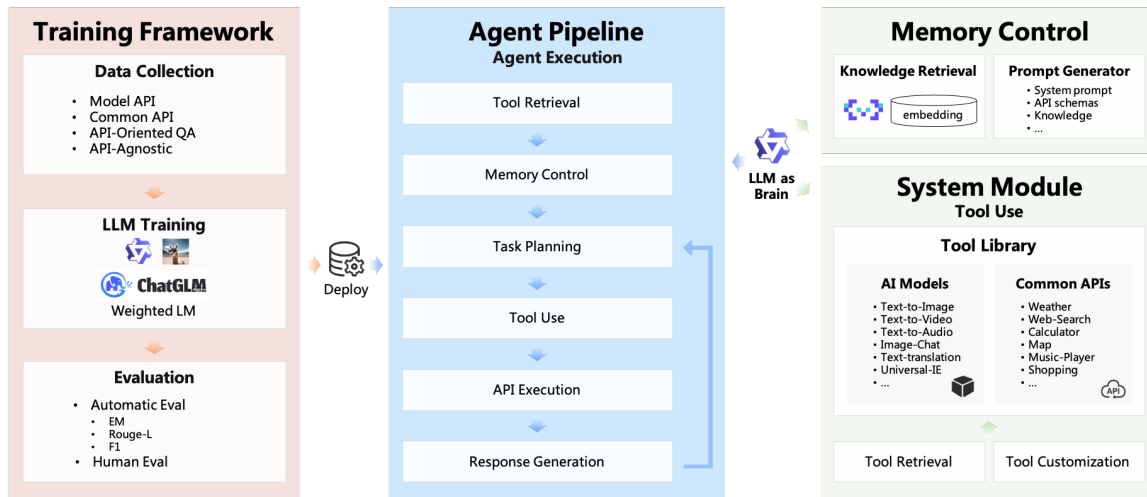


Figure 1: The overall system architecture of ModelScope-Agent.

can be done by simply setting the `model_name` and `model_config`. In `model_config`, the `model_id`, `model_revision`, and model parameter settings such as max sequence length, should be configured.

```
# LLM config "cfg_file"
from modelscope.utils.config import Config
model_cfg = Config.from_file(cfg_file)
llm = LocalLLM(model_name, model_cfg)
```

Furthermore, the ModelScope-Agent also provides a standard way to integrate new LLM. Users can add their own LLMs, by integrating the LLM pipeline into ModelScope. After that, the agent can select the new LLMs for training and inference.

## 2.2 Tool Use

**Tool Library** The tool library is used to configure and manage various collections of APIs used in the agent. ModelScope-Agent can support a wide range of both common APIs such as search APIs, and AI model APIs across NLP, CV, Audio and Multi-modal models in ModelScope and Hugging-Face. Each tool API consists of the API name, description, parameters and request functions. Users can easily choose and configure proper APIs in the library to build their own agents. The default APIs supported in the library can be referred to in Appendix A.1.

```
# tool default config file "default_file"
tool_cfg = Config.from_file(default_file)
```

**Register and Customize New Tool** The agent allows users to register and customize new tools, while also supporting quick integration of newly registered tools into the agent, enabling LLMs to selectively use the additional self-defined tools for specific applications. This can be simply done

by inheriting from a base class, namely `Tool`, and defining a new `CustomTool` with the API-related schema of API name, description, parameters, and request functions. More details about `CustomTool` can be referred to in Appendix A.2.

```
from modelscope_agent.tools import Tool
class CustomTool(Tool):
    # logic added here
    # refer example in Appendix A.2
tool_list = {'customo-tool': CustomTool()}
```

**Tool Retrieval and Execution** Due to the large amount of tool APIs in the tool library, a tool retrieval module is further introduced to recommend appropriate APIs for each instruction prompt. Specifically, we use the dense vector retrieval method based on the unified multilingual text-embedding API<sup>7</sup>. We vectorize both the text descriptions of the APIs and the instruction prompt using the text-embedding API. The top-3 most relevant APIs with the highest vector product scores are selected for tool use. As a result, the schema information of the retrieved APIs will be concatenated with other system prompts in the subsequent memory module and sent to LLMs as input. With the concatenated instruction prompt, the LLMs will plan and generate the API request, which will be executed by the agent. The agent will then return the results to the LLMs for continuous generation.

## 2.3 Memory Control

The memory module is used to retrieve and assemble a series of contextual information as input to the LLMs. It consists of a knowledge retrieval submodule and a prompt generator submodule, which are

<sup>7</sup><https://help.aliyun.com/zh/dashscope/getting-started-1>

responsible for external knowledge retrieval and instruction prompt generation, respectively.

**Knowledge Retrieval** It enables the agent to get access to up-to-date and localized information related with query prompt, thereby augmenting LLMs with dynamic and domain-specific knowledge. We follow the same dense vector retrieval method as the previous tool retrieval module and support large-scale knowledge retrieval from localized document corpus. Similarly, it allows users to customize by changing to other open-source retrieval frameworks.

**Prompt Generator** The prompt generator is used to assemble all available contextual information such as system prompt, API schema, retrieved knowledge, conversation history, and few-shot examples. According to the type of user query and the maximum length of the LLM, the users can selectively choose proper contextual information and assemble the required input to the LLM. In our agent, the prompt generator needs to be defined before the agent is constructed.

## 2.4 Agent Pipeline

In summary, we build the agent by combining all the modules: LLM controller, tool-use module, and memory module. With *agent.run*, the agent can efficiently execute and complete the instruction in a one-step generation. First, the agent retrieves query-related tools through the tool retrieval and combines the retrieved API schema with other contextual prompts in the memory module, to construct a new instruction prompt. Then, the agent sends this new prompt to the LLM, which plans whether and which API to call and generate an API request. Next, the agent will execute the selected API with the extracted API parameters and return the API results to the LLMs, which will continue to plan whether to call other APIs. If another API call is needed, the process is repeated, otherwise, the LLMs generate the final response and the agent returns the final result to the user.

```
agent = AgentExecutor(llm, tool_cfg,
                    additional_tool_list=tool_list)
agent.run("Draw a logo image of agent")
```

## 3 Training

### 3.1 Dataset

To facilitate building an agent with the ability to use tools while upholding an optimal level of user en-

gagement, we release a comprehensive tool dataset, MSAgent-Bench, utilizing ChatGPT synthetic data and the existing instruction-following datasets. Our released dataset encompasses 598k dialogues. Table 1 outlines the key differences between the released dataset and other publicly available tool learning datasets, while the data distribution of our dataset is illustrated in Figure 2. As demonstrated in the Table and Figure, we have made certain efforts to construct a comprehensive dataset that enables the effective training of an agent:

**Multilingual:** We collect instances in both Chinese and English, ensuring that the trained agent is capable of functioning in both languages.

**Various API Categories:** Our dataset supports Common APIs that have been registered by users or applied through online API platforms, as well as model APIs that can call neural models.

**Multi Turn Dialog:** In real-life scenarios, agents may need to request more specific clarification from users to complete a task or receive additional instructions after completing a previous task. Our dataset accounts for these scenarios and supports multi-turn user-agent interactions when using tools.

**API-Oriented QA:** An effective agent should possess knowledge of APIs. Our dataset incorporates API document QA tasks and task planning tasks which requires agents to offer appropriate suggestions to users on how to use various APIs to solve complex tasks.

**API-Agnostic Instructions:** To enhance the agent’s ability to follow common instructions and increase user engagement, we have incorporated both Chinese and English API-agnostic instructions within our dataset. These instructions place greater emphasis on the agent’s inherent capabilities rather than reliance on API invocation.

The data was collected by prompting ChatGPT (gpt-3.5-turbo) to generate instructions, API requests, and answers based on the API calling results, more details can be accessed in Appendix D.

### 3.2 Model Training

We use the MSAgent-Bench to fine-tune multiple open-source LLMs, including LLaMA (Touvron et al., 2023), Qwen (QwenLM, 2023), ChatPLUG (Tian et al., 2023) etc. We train all the open-source LLMs in a multi-round conversation mode and concatenate all the prompts and answers. Compared to common instruction tuning data, the tool learning samples focus more heavily on the

| Dataset                        | Language          | Instance Type          | # Instances | API type               | Avg. Turn | Avg. Step |
|--------------------------------|-------------------|------------------------|-------------|------------------------|-----------|-----------|
| API-Bank (Li et al., 2023)     | English           | Tool Use               | 264         | Common API             | 3.27      | 1.92      |
| ToolAlpaca (Tang et al., 2023) | English           | Tool Use               | 3.9 K       | Common API             | 1         | 1.66      |
| Gorilla (Patil et al., 2023)   | English           | Tool Use               | 16.4 k      | Model API              | 1         | 1         |
| GPT4Tools (Yang et al., 2023)  | English           | Tool Use               | 71.4 K      | Model API              | 1         | 1         |
| ToolBench (Qin et al., 2023)   | English           | Tool Use               | 26.9 K      | Common API             | 1         | 4.1       |
| MSAgent-Bench (ours)           | English + Chinese | Tool Use + Common Chat | 598 K       | Common API + Model API | 1.52      | 1.31      |

Table 1: The statistics of MSAgent-Bench and other existing tool learning datasets.

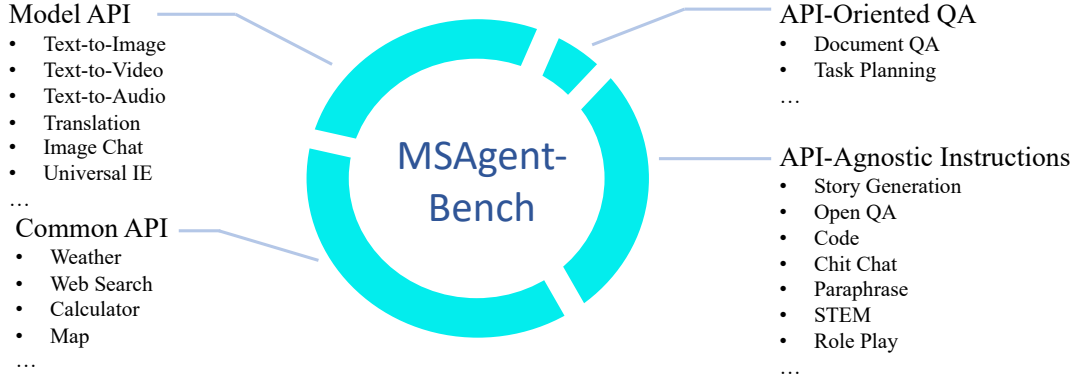


Figure 2: The instance types and distribution of our collected MSAgent-Bench.

accuracy of tool selection and API parameter prediction. Therefore, we propose a simple training strategy, Weighted LM, which enhances the training of generation of API names and parameters, while zero-out the loss of tokens from the user prompt and the tool execution. More details can be referred to in Appendix B.3.

```
kwargs = dict(model=model, ...)
trainer: EpochBasedTrainer = build_trainer
(name=args.trainer, default_args=kwargs)
trainer.train()
```

## 4 Evaluation

Our evaluation system, MSAgent-Eval, comprises two modules: an automatic evaluation framework that comprehensively evaluates the API usability of the agents and a human evaluation framework implemented by an agent arena that reflects the preferences of human users.

### 4.1 Automatic Evaluation Framework

In automatic evaluation, we mainly focus on evaluating the agent’s ability to generate accurate API requests and the proper answers according to the API calling results. Specifically, we use the action exactly match score (Action EM) which measures whether the agent uses the correct API as the reference gold API, and the ROUGE-L score which measures the similarity between the generated response and the gold answer. Additionally, we intro-

duce a novel metric called Argument F1 for fully evaluating the quality of API requests. To compute Argument F1, we categorize the arguments in the agent’s API request into two cases, namely Half match (HM) and Full match (FM), representing the correct argument but with the wrong value and the correct argument with the correct value, respectively. Suppose the gold argument number in the API is  $|A|$ , and the number of arguments in the agent API request is  $|A^*|$ , we compute the new Recall and Precision as follows:

$$R = (0.5 \times \# \text{HM} + \# \text{FM}) / |A| \quad (1)$$

$$P = (0.5 \times \# \text{HM} + \# \text{FM}) / |A^*| \quad (2)$$

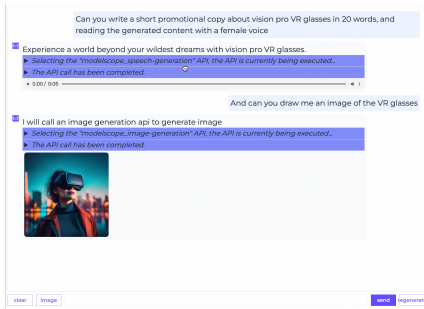
and the final argument F1 is computed as:

$$F1 = 2(R * P) / (R + P). \quad (3)$$

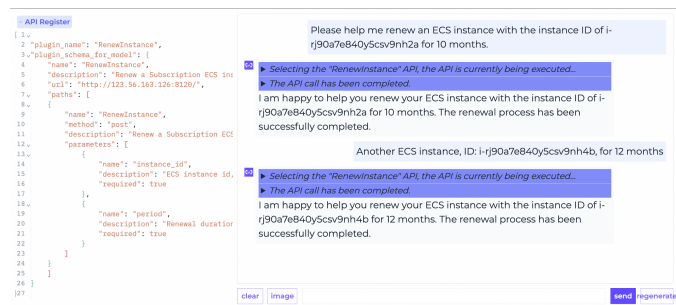
A sample code for the automated evaluation of agents is provided below:

```
from tool_agent_finetune import evaluation
EM, F1, ROUGE = evaluation(refs, preds)
```

Expert annotators were engaged to annotate the evaluation instances, with the task of providing diverse instructions, manually documenting correct API calling requests, and writing appropriate responses. The statistics of our currently assembled test data is in Appendix B.1, and the automatic evaluation scores of our trained agents can be found in Appendix B.2. We also guarantee the



(a) ModelScope Intelligent Assistant



(b) Register and Use New Tools on Alibaba Cloud

Figure 3: Demo cases of ModelScopeGPT based on ModelScope-Agent .

users to upload their own annotated test examples to accurately evaluate the performance of agents in customized scenarios.

## 4.2 Human Evaluation with Agent Arena

Inspired by the Arena for ChatBots (Zheng et al., 2023), we have built an accessible Agent Arena<sup>8</sup> that allows users to furnish instructions to two anonymous agents, based on the provided APIs. Subsequently, users have the opportunity to vote on which Agent performs better in tackling the instruction with the given APIs. In accordance with the framework presented by Zheng et al. (2023), we adopt a system of ELO ratings and leaderboard maintenance for the participating Agents.

## 5 Usage Example of ModelScopeGPT

In this section, we showcase a successful application of ModelScope Community, ModelScopeGPT<sup>9</sup>, based on our ModelScope-Agent.

**ModelScope Intelligent Assistant** Based on ModelScope-Agent, we have developed an intelligent assistant for the ModelScope Community, namely ModelScopeGPT. It uses LLMs as a controller to connect dozens of domain-specific AI models in the ModelScope open-source community, covering NLP, CV, Audio, and Multi-Modal fields. To make the pipeline more practical, we have included API retrieval and knowledge retrieval tools to automatically select proper APIs and get access to the local ModelScope knowledge. As shown in Figure 3a, ModelScopeGPT can support API calls in multi-turn conversations and generate correct API call parameters using information from

previous conversations. More cases can refer to Appendix C. As a result, ModelScopeGPT has achieved a total request number of over 170k from 40k user visits within one month after its release.

**Register and Use New Tools** Another key feature of an agent is its generalization capability to unseen APIs. This allows users to quickly register their own APIs and customize their specific applications. Therefore, we test the generalization ability of ModelScopeGPT by applying it to an Alibaba Cloud application scenario. As shown in Figure 3b, we first found an API for renewing an ECS instance on Alibaba Cloud. Then, we registered the API schema defined in the tool library to the agent. Finally, we entered the prompt "Please help me renew an ECS..." in the demo. The agent generated a request through planning, selected the appropriate API, called the API to renew the instance successfully, and provided a reply to inform the user that the renewal was completed. This test demonstrates that the open-source LLM optimized based on the released API dataset has a strong generalization ability towards unseen APIs.

## 6 Conclusion

ModelScope-Agent aims to facilitate building AI Agent applications and research based on open-source LLMs by providing a general and customizable agent framework covering flexible system design, data collection, model training, evaluation and usage examples in real-world applications. It provides an open-source, community-driven library for AI Agent learning and best practices for building an agent system with open-source LLMs. We hope ModelScope-Agent can help pave the way towards a new era of AI Agent.

<sup>8</sup><https://modelscope.cn/studios/LLMZOO/Chinese-Arena/summary>

<sup>9</sup><https://modelscope.cn/studios/damo/ModelScopeGPT/summary>

## Ethics Statement

**Intended Use.** ModelScope-Agent is designed to facilitate building AI Agent applications and research based on open-source LLMs, by providing a general and customizable agent system.

**Potential Misuse.** Although we have only trained with the tool-use datasets and gone through certain data filtering rules, it is still possible that the customized model may generate some biased, fake, and unsafe information. Our agent framework also provides users with the freedom to select proper LLMs and upload their own clean data for training. It is also important to design specific methods to improve the safety of the agent framework in the future.

## References

- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. *Do as i can, not as i say: Grounding language in robotic affordances*. *arXiv preprint arXiv:2204.01691*.
- Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, et al. 2023. Falcon-40b: an open large language model with state-of-the-art performance.
- 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*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pilla, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. *Palm: Scaling language modeling with pathways*. *CoRR*, abs/2204.02311.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. *Training compute-optimal large language models*. *arXiv preprint arXiv:2203.15556*.
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. 2023. *Inner monologue: Embodied reasoning through planning with language models*. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR.
- Huggingface. 2023. Transformers agent. Website. [https://huggingface.co/docs/transformers/transformers\\_agents](https://huggingface.co/docs/transformers/transformers_agents).
- Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. *Api-bank: A benchmark for tool-augmented llms*. *arXiv preprint arXiv:2304.08244*.
- 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*.
- OpenAI. 2022. *Chatgpt: Optimizing language models for dialogue*.
- OpenAI. 2023. *GPT-4 technical report*. *CoRR*, abs/2303.08774.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. *Gorilla: Large language*



- model connected with massive apis. *arXiv preprint arXiv:2305.15334*.
- Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. [Tool learning with foundation models](#). *arXiv preprint arXiv:2304.08354*.
- QwenLM. 2023. [Qwen-7b](#).
- 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*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). *arXiv preprint arXiv:2302.04761*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face](#). *arXiv preprint arXiv:2303.17580*.
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. [Toolalpaca: Generalized tool learning for language models with 3000 simulated cases](#). *arXiv preprint arXiv:2306.05301*.
- THUDM. 2023. [Chatglm](https://github.com/THUDM/ChatGLM-6B). <https://github.com/THUDM/ChatGLM-6B>.
- Junfeng Tian, Hehong Chen, Guohai Xu, Ming Yan, Xing Gao, Jianhai Zhang, Chenliang Li, Jiayi Liu, Wenshen Xu, Haiyang Xu, Qi Qian, Wei Wang, Qinghao Ye, Jiejing Zhang, Ji Zhang, Fei Huang, and Jingren Zhou. 2023. [Chatplug: Open-domain generative dialogue system with internet-augmented instruction tuning for digital human](#). *arXiv preprint arXiv:2304.07849*.
- 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](#). *arXiv preprint arXiv:2302.13971*.
- Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. [Visual chatgpt: Talking, drawing and editing with visual foundation models](#). *arXiv preprint arXiv:2303.04671*.
- Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. [Gpt4tools: Teaching large language model to use tools via self-instruction](#). *arXiv preprint arXiv:2305.18752*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). *arXiv preprint arXiv:2306.05685*.

## A Library

### A.1 Tool List

| API Name (language)    | Description                         | Type       |
|------------------------|-------------------------------------|------------|
| Text-to-Image(en)      | Converts text to an image.          | Model API  |
| Text-to-Image(zh)      | Converts text to an image.          | Model API  |
| Text-to-Video(en)      | Converts text to a video.           | Model API  |
| Text-to-Audio(en)      | Converts text to audio.             | Model API  |
| Text-to-Audio(zh)      | Converts text to audio.             | Model API  |
| Image-Chat(en)         | Image chat.                         | Model API  |
| Translation-zh2en      | Translates Chinese text to English. | Model API  |
| Translation-en2zh      | Translates English text to Chinese. | Model API  |
| Universal-IE(zh)       | Extracts structured information.    | Model API  |
| Text-to-Geographic(zh) | Extracts geographic information.    | Model API  |
| NER(zh)                | Recognizes named entities in text.  | Model API  |
| API-Retrieval          | Retrieves relevant APIs             | Common API |
| ModelScope-Retrieval   | Retrieves modelscope docs.          | Common API |

Table 2: The statistics of default tool list. Supported input languages for the APIs are listed in parentheses.

### A.2 CustomTool

User can customize their own tools by inheriting a base tool and defining the tool names, descriptions, and parameters according to a pre-defined schema. Moreover, you can implement `_local_call()` or `_remote_call()` depending on your specific requirements. To illustrate, below is an example of a custom tool:

```
class CustomTool(Tool):
    description = 'xxx'
    name = 'xxx'
    parameters: list = [{
        'name': 'xxx',
        'description': 'xxx',
        'required': True
    }]

    def _local_call():
        ...

    def _remote_call():
        ...
```

## B Experiment Setup

### B.1 Evaluation Benchmark

To assess the generalization of the trained agent, we include 10 in-domain APIs that appear in the training set of ModelScope-Agent and 10 real unseen APIs<sup>10</sup>. We also account for the multi-turn ability of the agent by annotating several multi-turn scenarios in our evaluation benchmark. Our test instances were annotated by asking the human experts to write diverse instructions first. Then the human experts were ask to write the JSON API request and answer the instructions properly after obtaining the API calling results. Our final testing

<sup>10</sup>In progress, we will include more APIs in the future.

dataset consisted of 360 conversations with 2059 text snippets as the references to be compared with the agent prediction, which comprise 798 API requests and 1261 plain text answers according to the previous calling results.

### B.2 Evaluation Results

| Model             | ROUGE-L | Action EM | Argument F1 |
|-------------------|---------|-----------|-------------|
| ChatGPT (2-shot)* | 36.70   | 34.82     | 25.51       |
| LLaMA             | 39.16   | 58.60     | 44.98       |
| ChatPLUG          | 46.45   | 68.29     | 55.12       |
| MSAgent-7B        | 51.35   | 87.23     | 68.09       |

Table 3: Automatic evaluation results. \* represents that we do not fine-tune ChatGPT but use in-context learning with 2 demonstrations.

We compare the models trained in our proposed ModelScopeGPT. The automatic evaluation results are shown in Table 3. Based on the findings obtained from our experimentation, it is evident that ChatGPT with in-context learning yielded inferior results as compared to other models that were subjected to finetuning. Furthermore, LLaMA underperformed when compared to other fine-tuned models. Our error study revealed that the lower performance of ChatGPT and LLaMA could be attributed to a large proportion of Chinese test cases in our test set. The models (ChatPLUG, MSAgent-7B<sup>11</sup>) that performed better were those that predominantly focused on Chinese data. Our investigation revealed that ChatGPT and LLaMA exhibited limitations in user intent recognition, which ultimately led to their suboptimal performance on Action EM. Among the models examined, MSAgent-7B displayed the most favorable performance, which could be attributed to the superior performance of its basic model.

### B.3 Weighted LM

We give an example of the training strategy Weighted LM. As show in Figure 4, tokens with different colors have different loss weights. For the user input prompt, we set the loss weight to 0, so that the model does not calculate the loss for the prompt. For the API-Agnostic text of the assistant, we keep the loss weight as 1. Finally, for the important text of the API calling, such as API name, parameters, URL, etc., we set the loss weight to 2, which can improve the generation accuracy of API calling.

<sup>11</sup><https://modelscope.cn/models/damo/ModelScope-Agent-7B>

```

User: Write a brief introduction about VR glasses in 10 words
Assistant: Wearable devices for immersive virtual reality experience.
User: read it out loud in a female voice.
Assistant: <startofthink> { "api_name": "modelscope_speech-generation", "url": "http://33.57.174.141:5000/damo/speech_sambert-hifigan_tts_zh-cn_16k", "parameters": {"text": "Wearable devices for immersive virtual reality experience.", "gender": "woman"}} <endofthink>
<starofexec> { "result": " <audio id=audio controls=preload=none> <source id=wav src=ursfe2.wav> </audio> " } <endofexec>
<audio id=audio controls=preload=none> <source id=wav src=ursfe2.wav> </audio>

```

Loss weight

- 0.0
- 1.0
- 2.0

Figure 4: Example of training strategy for weighted LM. Different colored tokens have different loss weights.

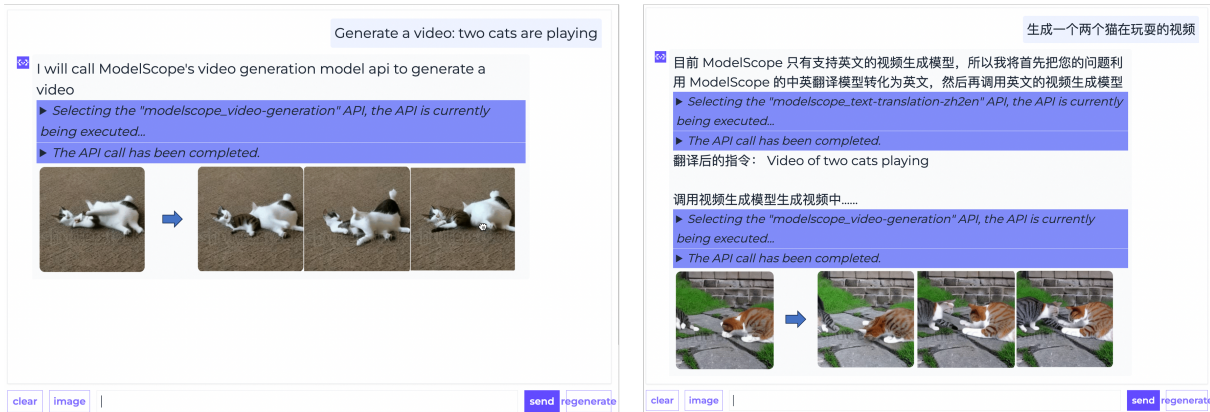


Figure 5: Single-step tool-use instructions, text-to-video cases. We have captured a few frames of the video to display. Testing the model using the same semantic instruction in both English (left) and Chinese (right).

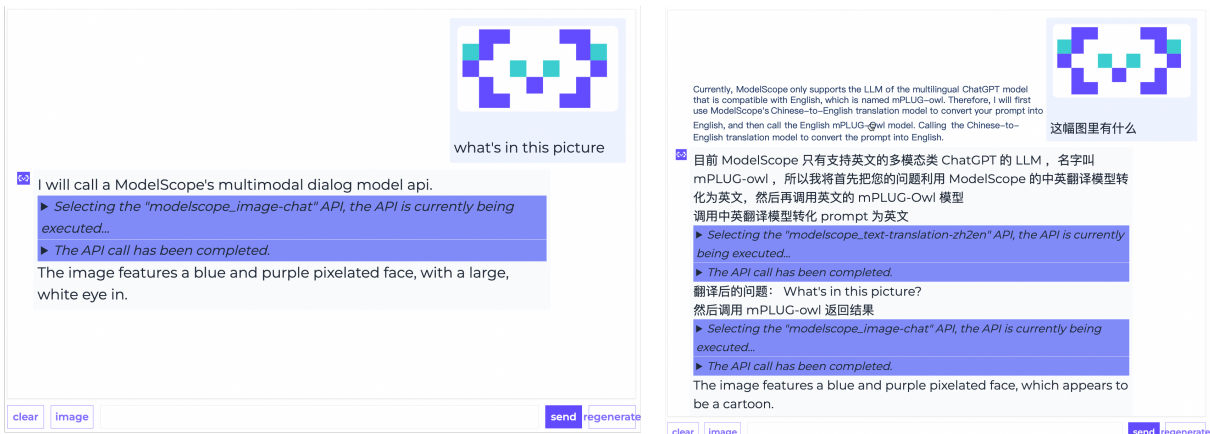


Figure 6: Single-step tool-use instructions, image-chat cases. Testing the model using the same semantic instruction in both English (left) and Chinese (right).

## C Cases

In this section, we show the qualitative results about ModelScopeGPT implementation based on ModelScope-Agent.

**Single-step Tool Use** As shown in Figure 5 and 6, the instruction expects the model to generate a video and chat about the image respectively. These instructions can be completed with a single step of tool use.

**Multi-step Tool Use** As shown in Figure 7, the instruction expects the model to write the promo-

tional copy first, then read it, and finally generate a video. These instructions require the model to have the ability of multi-step Tool use. In the Chinese case, our model accurately completed the three-step tool use.

**Multi-turn Tool Use** As shown in Figure 8, the instruction requires the model to have the ability to multi-turn conversation and use the history conversation. Our model can accurately call the API and capture the content of the previous conversation to generate API parameters.

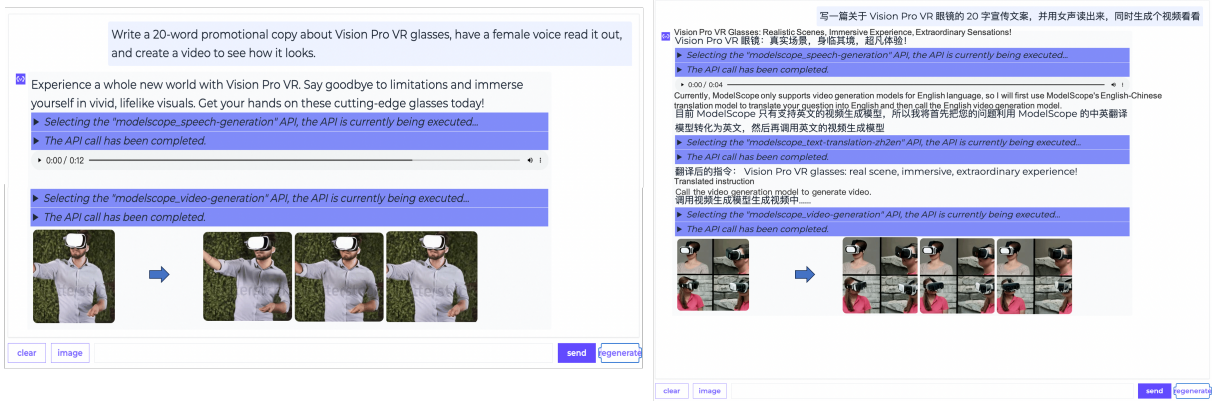


Figure 7: Multi-step tool-use instructions. We have captured a few frames of the video to display. Testing the model using the same semantic instruction in both English(left) and Chinese(right).

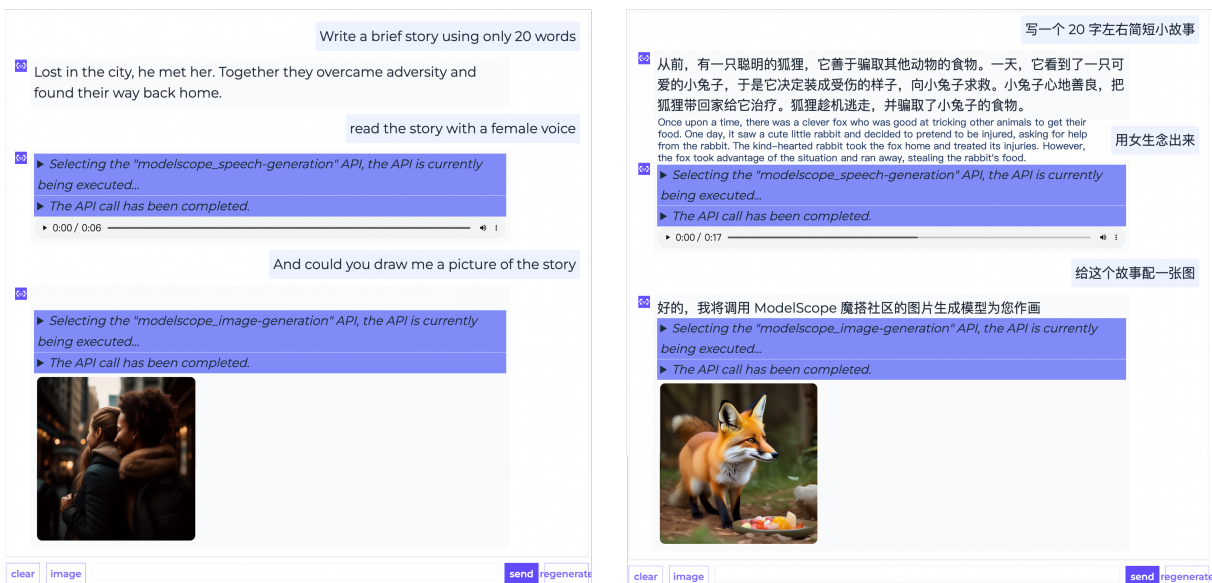


Figure 8: Multi-turn tool-use instructions, text-to-speech and text-to-image cases. Testing the model using the same semantic instruction in both English(left) and Chinese(right).



Figure 9: Multi-turn tool-use instructions, text-to-speech and text-to-image cases. Testing the model using the same semantic instruction in both English(left) and Chinese(right).

**In-domain Knowledge QA** As shown in Figure 9, the instruction requires the model to retrieve in-domain knowledge and use the retrieved knowledge

to answer questions.

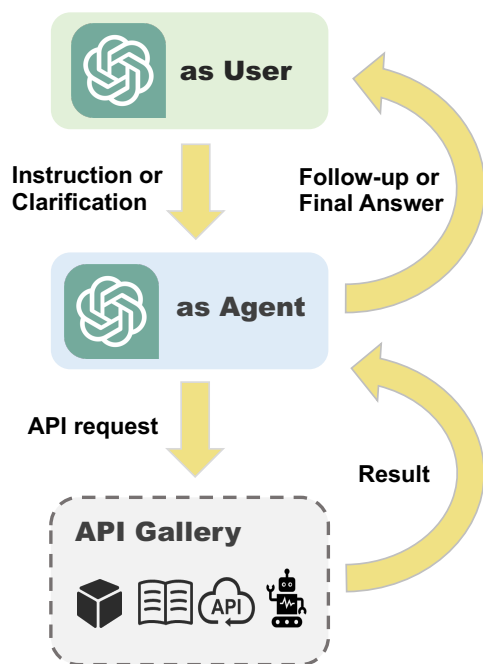


Figure 10: The data collection procedure of MSAgent-Bench.

## D Data Collection Procedure

We collected our dataset by using prompt engineer to simulate the agent scenarios with two ChatGPTs (gpt-3.5-turbo). One of the ChatGPTs was prompted to act as the user, while the other was assigned to act as the agent. In order to expand the domains and functionalities of APIs presented in the training data, rather than the existing real APIs, we also included a number of synthetic APIs that were generated by ChatGPT. When these synthetic APIs were incorporated into the dialogues, we prompted another ChatGPT to serve as the API and return the relevant calling outcomes.

The data collection procedure is shown in Figure 10. Initially, a set of random in-context demonstrations were provided to ChatGPT for generating an instruction. This instruction could either be a regular one or one that requires solving with APIs, depending on the demonstrations provided. Subsequently, ChatGPT was prompted to act as an agent by first thinking about which action to undertake. If no API calls were deemed necessary, or if the user clarification is needed, the agent would respond with a follow-up response to the user. Otherwise the agent will send API request to the API gallery. After receiving the result of the API call, the agent would assess the situation and decide on the next action. This iterative process of the "user-agent-API"

loop would continue until the agent determined that it was appropriate to terminate the conversation with the final answer. After acquiring the raw dataset, we applied filtering mechanisms to eliminate instances in which ChatGPT generated API requests containing hallucinated API names and parameters that were absent from the retrieved API. Additionally, we excluded instances in which ChatGPT generated illegal API requests, thus resulting in a refined and finalized dataset.

As introduced in Section 3.1, we collect instances across different languages and topics, the detailed statistics of our collected data are shown in Table 4.

| Instance Type            | # Instances |
|--------------------------|-------------|
| Chinese                  | 532,436     |
| English                  | 66,444      |
| Common API               | 211,026     |
| Model API                | 58,338      |
| API-Oriented QA          | 5,000       |
| API-Agnostic Instruction | 329,776     |

Table 4: The statistics of our collected dataset.

## E Related Work

### E.1 Large Language Models

Recent years have witnessed rapid development in the field of Large Language Models (LLMs). Typical models, such as GPT3 (Brown et al., 2020), Gopher (Rae et al., 2021), Chinchilla (Hoffmann et al., 2022), PaLM (Chowdhery et al., 2022) and LLaMA (Touvron et al., 2023), have shown impressive zero and few-shot generalization abilities on a wide range of NLP tasks, by scaling up the model and data size. A remarkable milestone is the release of ChatGPT (OpenAI, 2022) or GPT4 (OpenAI, 2023), which has greatly revolutionized the paradigm of AI development. As a result, a rising trend of open-source LLMs has emerged to challenge and catch up their closed-source counterparts like ChatGPT and Claude, such as BLOOM (Muenighoff et al., 2022), LLaMA (Touvron et al., 2023), Falcon (Almazrouei et al., 2023), ChatGLM (THUDM, 2023). Despite the great breakthrough, LLMs are trained as text generators over plain text corpora, thus performing less well on other tasks such as multi-modal tasks. It also falls short on tasks that require up-to-date information, which are beyond the pretraining data. Using tools or external APIs can help overcome the limitations and harness the power of LLMs to facilitate seam-

less connections with downstream applications. In ModelScope-Agent, we provide the whole customizable framework and best practices for building an agent system, which enables open-source LLMs to use tools and external APIs.

## E.2 Agent & Tool Learning

The utilization of Large Language Models (LLMs) as a controller to construct an agent system has emerged as a prominent research area. Several related works employ prompt engineering techniques on closed-source LLMs, such as ChatGPT (OpenAI, 2022) and Claude, to enable their application in specific domains. For instance, VisualChatGPT (Wu et al., 2023) and HuggingGPT (Shen et al., 2023) facilitate the HuggingFace model callings accessible to OpenAI LLMs. SayCan (Ahn et al., 2022) and inner monologue (Huang et al., 2023) integrate LLMs with robots to achieve robotic systems. Notably, recent works such as Langchain and Auto-GPT encompass a wide range of tools, including common APIs and neural models, and enhance long-term reasoning and human-agent interaction whilst solving tasks, which demonstrate the immense potential for building a generalized agent.

Numerous endeavors have also been made to enable open-source LLMs to utilize tools. For instance, Gorilla (Patil et al., 2023) and GPT4Tools (Yang et al., 2023) generate training data using self-instruction techniques to train open-source LLMs to effectively utilize neural models. ToolAlpaca (Tang et al., 2023) and ToolLLaMA (Qin et al., 2023) train LLAMA using common APIs, with the distinction that ToolAlpaca employs synthetic APIs from LLMS, whereas ToolLLaMA utilizes real APIs.

Overall, compared to the above-mentioned methods, ModelScope-Agent differs in the following aspects. Firstly, our method includes a universal training framework that supports user-customized agent learning for open-source models to meet industrial needs. Secondly, ModelScope-Agent can support various APIs in different fields, including model APIs and common APIs, while previous works only support certain specific APIs.

## F Future Work

In the future, we will evolve to support more sophisticated agent architectures such as ReAct and code interpreter. In the meantime, we will

continuously improve the capabilities required by open-source LLMs as agents. ModelScope-Agent relies on the ModelScope community and will adapt to more new open-source LLMs in the future, providing more applications developed based on ModelScope-Agent, such as personal-assistant-agent, story-agent, motion agent, and so on.

# EffOCR: An Extensible, Open-Source Package for Efficiently Digitizing World Knowledge

Tom Bryan<sup>1</sup>, Jacob Carlson<sup>1</sup>, Abhishek Arora<sup>1</sup>, Melissa Dell<sup>1,2\*</sup>

<sup>1</sup> Harvard University; Cambridge, MA, USA.

<sup>2</sup> National Bureau of Economic Research; Cambridge, MA, USA.

All authors contributed equally. \*Corresponding author: melissadell@fas.harvard.edu.

## Abstract

Billions of public domain documents remain trapped in hard copy or lack an accurate digitization. Modern natural language processing methods cannot be used to index, retrieve, and summarize their texts; conduct computational textual analyses; or extract information for statistical analyses, and these texts cannot be incorporated into language model training. Given the diversity and sheer quantity of public domain texts, liberating them at scale requires optical character recognition (OCR) that is accurate, extremely cheap to deploy, and sample-efficient to customize to novel collections, languages, and character sets. Existing OCR engines, largely designed for small-scale commercial applications in high resource languages, often fall short of these requirements. EffOCR (EfficientOCR), a novel open-source OCR package, meets both the computational and sample efficiency requirements for liberating texts at scale by abandoning the sequence-to-sequence architecture typically used for OCR, which takes representations from a learned vision model as inputs to a learned language model. Instead, EffOCR models OCR as a character or word-level image retrieval problem. EffOCR is cheap and sample efficient to train, as the model only needs to learn characters' visual appearance and not how they are used in sequence to form language. Models in the EffOCR model zoo can be deployed off-the-shelf with only a few lines of code and include lightweight models designed for mobile phones that are extremely cheap to deploy. Importantly, EffOCR also allows for easy, sample efficient customization with a simple model training interface and minimal labeling requirements due to its sample efficiency. We illustrate the utility of EffOCR by cheaply and accurately digitizing 20 million historical U.S. newspaper scans, evaluating zero-shot performance on randomly selected documents from the U.S. National Archives, and accurately digitizing a Japanese document collection for which all other OCR solutions failed.

## 1 Introduction

Vast document collections remain trapped in hard copy or lack accurately digitized texts. For example, the U.S. National Archives holds approximately 13.28 billion pages of textual records, most of which are in the public domain.<sup>1</sup> These documents are preserved because they are central to the workings of the U.S. government, have long-term research value, or provide valuable information for the public, but working with most of them is costly and time-consuming. The U.S. National Archives are not unique: many other countries have national archives with public domain collections numbering in the billions of pages, not to mention state and local archives and libraries. Without accurate machine-readable data, modern natural language processing (NLP) tools cannot be used to index, retrieve, and summarize materials; conduct computational textual analyses; or extract information for statistical investigations. Public domain texts, if accurately digitized, could also provide massive scale information for training large language models, with no risks of copyright infringement.

Using optical character recognition (OCR) to digitize public domain collections on a large scale entails several challenges.

**Cost:** First, the OCR solution must be cheap to deploy, given document collections whose size numbers in the millions or even billions of pages. Commercial engines - as well as large open-source OCR models - fall well short of this requirement. Using them to digitize large-scale collections would require astronomical budgets.

**Accuracy:** Second, digitized texts need to

<sup>1</sup>For documents published in the United States, the public domain includes any content published by a U.S. government officer/employee in the course of official duties, all content published more than 95 years ago, and some content published before 1989 that either wasn't published with a notice or did not renew copyright. This is common, for instance, in the case of publications like local newspapers (Ockerbloom, 2019). See the supplementary materials for details.

be sufficiently accurate for end users' objectives, which are highly diverse. Accuracy can be particularly central for quantitative applications, for which small errors can create major statistical outliers. Models for lower resource languages, if they exist, tend to perform much worse than models for high resource settings like English.

**Sample efficient, easy training:** Documents are highly heterogeneous in terms of their fonts or handwritings, languages, scripts, backgrounds, and artifacts from scanning and aging. There are a diversity of documents for which no existing OCR solution works zero-shot, particularly in low resource languages. Yet stakeholders who would like to digitize these documents rarely have familiarity with deep learning frameworks. Bringing high quality OCR to low resource settings requires a simple API for training and a sample efficient architecture, with an accessible compute and annotation burden.

**A diversity of pre-trained and tuneable models:** Users have diverse accuracy needs, scaling requirements, and budgets. A comprehensive OCR solution would make it easy to compare the accuracy and deployment costs of models of varying sizes so that users can choose the one that best suits their needs for a particular application.

To meet these objectives, we developed EffOCR, an open-source OCR package designed for researchers, libraries, and archives seeking a computationally and sample efficient OCR solution for digitizing diverse document collections. EffOCR has two key ingredients: 1) a novel OCR architecture and 2) a carefully designed interface to facilitate off-the-shelf OCR usage, customization via model training if necessary, and easy sharing of OCR models.

The novel EffOCR model architecture is treated in detail in [Carlson et al. \(2023\)](#), where we compare accuracy, sample efficiency, and deployment costs to a range of popular OCR engines. In short, OCR predominantly models text recognition as a sequence-to-sequence (seq2seq) problem, in which learned representations from a vision model are taken as inputs to a learned language model. Learning how vision embeddings are used in sequence to form language requires large amounts of data. For example, the predominant transformer sequence-to-sequence OCR package ([Li et al., 2021](#)) was trained on 684 million text lines using 32 32GB V100 GPU cards. State-of-the-art seq2seq OCR is sample-inefficient to tune and infeasible for users

to extend to low resource languages, which may not even have a transformer large language model (LLM) that can be used to initialize the model, as language modeling advances are concentrated in less than two dozen languages ([Joshi et al., 2020](#)). The typical stakeholder working with low resource documents has a minimal budget for training and limited experience with deep learning frameworks, underscoring the need for a much more sample efficient framework with an easy-to-use API.

Additionally, seq2seq OCR requires autoregressive decoding, which makes inference slower than it would be, all else equal, with parallel decoding.

EffOCR abandons the seq2seq OCR model that predominates in the literature, instead modeling OCR as a word or character level *image* retrieval problem. EffOCR first localizes words using highly accurate, scalable object detection methods ([Ultalytics, 2023](#); [Chen et al., 2019](#); [Wu et al., 2019](#)). Recognition is then modeled as a contrastively trained image retrieval problem, where image embeddings of the same character or word have similar representations, regardless of their style. EffOCR is trained primarily on digital fonts, combined with a modest number of character and word crops from real-world documents. At inference time, characters/words are recognized by computing their nearest neighbor in an offline dictionary of exemplar embeddings created with a digital font. [Carlson et al. \(2023\)](#) show, using English, Japanese, and Polytonic Greek benchmarks, that the EffOCR architecture is accurate, highly sample efficient, cheap to train, and extremely fast to deploy when using backbones designed for mobile phones.

To meet the challenges of digitizing large-scale and low-resource document collections, the EffOCR package contains the following components:

1. An off-the-shelf toolkit for applying OCR models with just a few lines of code
2. A repository of pre-trained OCR models that underlies off-the-shelf usage
3. ONNX runtime support for fast deployment
4. Comprehensive tools for efficient model tuning
5. Supports models from popular backends ([Chen et al., 2019](#); [Ultalytics, 2023](#)) for initializing localization and any *timm*-supported model for initializing recognition



6. Easy sharing of models, to promote reusability, reproducibility, and extensibility

EffOCR has been extensively tested. For example, we have used it to cheaply digitize 20 million pages of historical public domain U.S. newspaper scans that are extremely heterogeneous, posting the massive-scale output to Hugging Face.<sup>2</sup> Creating this dataset within our modest budget while meeting accuracy requirements would have been impossible without EffOCR. We have also examined performance in settings where no existing OCR solutions provide usable output, and tested zero-shot performance on a random selection of U.S. National Archive documents, with a model that did not see any similar content during training. Tutorials are available at <https://effocr.github.io/>.

EffOCR has a GNU General Public License. It is being actively maintained and crowd-sourcing of annotations to expand the pre-trained model zoo to other languages and settings, including handwriting, is underway.

The rest of this paper is organized as follows. Section 2 briefly compares EffOCR to existing, popular OCR solutions. Section 3 describes the key features of the OCR package, and Section 4 examines several use cases: using EffOCR to digitize 20 million historical newspaper scans, using EffOCR zero-shot on randomly selected collections from the U.S. National Archives, and using EffOCR to digitize a historical Japanese publication for which all existing OCR solutions fail. Finally, Section 5 discusses the limitations of the EffOCR package.

## 2 Comparisons to Other OCR Engines

There is a vast literature on OCR. Of primary interest here are widely used OCR softwares, which are the most plausible alternatives to EffOCR.

EffOCR- as the name suggests - is tailored towards applications requiring computational or sample efficiency. Carlson et al. (2023) conduct detailed experiments comparing the EffOCR architecture to other widely used solutions, considering accuracy, sample efficiency, and computational efficiency. We refer the interested reader to that paper for details, summarizing the two key themes that emerge here.

**Customization is highly relevant:** As the preponderance of researchers still using data entry

<sup>2</sup><https://huggingface.co/datasets/dell-research-harvard/AmericanStories/>

firms suggests, sometimes no existing OCR solution provides acceptable accuracy. For typewritten Japanese documents from the mid-20th century, that are of considerable relevance to studying Japan’s remarkable 20th century growth performance, Carlson et al. (2023) show that the best performing engine (Baidu, the leading commercial OCR for Asian languages) gets over half of characters wrong. The widespread failure of OCR to provide acceptable results is also evidenced by a large post-OCR error correction literature (e.g., Lyu et al. (2021); Nguyen et al. (2021); van Strien et al. (2020)).

EffOCR is significantly more sample efficient than leading open-source OCR engines: EasyOCR (JaidedAI, 2021), TrOCR (Li et al., 2021), and PaddleOCR (Du et al., 2022), as shown in the supplementary materials.<sup>3</sup> Learning to recognize the visual features of individual characters is a highly parsimonious problem, making EffOCR cheap to tune or train from scratch. Because EffOCR does not need to understand language, it is straightforward to extend to new languages and scripts, including those that lack a transformer large language model to initialize state-of-the-art seq2seq. The convolutional models in the EffOCR model zoo can be trained on a Google Colab account, whereas training TrOCR on 684 million text lines required 32 32GB V100 cards.

A central aim of EffOCR is to democratize access to OCR to low resource languages and settings that are difficult to study because existing solutions are not suitable to these use cases. While we do not have the resources to train OCR models for all these settings, our simple APIs for training models and uploading them to the EffOCR model hub can encourage the crowdsourcing of this effort.

**The most accurate OCR engines in high resource settings (e.g., English) are costly to deploy at scale:** TrOCR (Base) is a highly accurate state-of-the-art English OCR. With 334 million parameters, it is nearly 50 times slower to deploy than our pre-trained lightweight EffOCR English word recognition model, while offering only relatively modest gains on the evaluation tasks in Carlson

<sup>3</sup>EasyOCR uses a seq2seq convolutional recurrent neural network (CRNN) framework (Shi et al., 2016), TrOCR uses a seq2seq encoder-decoder transformer (Li et al., 2021), and PaddleOCR’s uses Single Vision Text Recognition (SVTR), which like EffOCR abandons seq2seq, dividing text images into small (non-character) patches, using mixing blocks to perceive inter- and intra-character patterns, and recognizing text by linear prediction (Du et al., 2022).

et al. (2023).<sup>4</sup> For English, Google Cloud Vision (GCV) - a proprietary commercial product - dominated all open-source solutions (including EffOCR), but would have been orders of magnitude more costly to deploy. In our experience, it is frequently outside academic budgets for larger projects.

Lightweight EffOCR models are also faster than Tesseract and PaddleOCR - with the comparison to EasyOCR depending on the hardware used for deployment. This is despite having around 8x more parameters than Tesseract and around 4x more than EasyOCR (parameter count is similar to PaddleOCR). This is achieved through parallel rather than sequential decoding and ONNX integration. EffOCR is also significantly more accurate on tasks like digitizing the 20 million U.S. historical newspaper scans.

Users for whom neither computational nor sample efficiency is of concern - because they are working in a well-resourced context and don't face cost constraints for the scale of their problem - are not our target audience and may well find an existing OCR engine like Google Cloud Vision better meets their needs. In practice, academic or large-scale archival digitization of document collections often involves low-resource languages or settings, tight budget constraints, or both.

### 3 The EffOCR Library

#### 3.1 Off-the-shelf Usage

At the core of EffOCR is an off-the-shelf toolkit. EffOCR is a modular framework, that first localizes lines, characters, and (for some models) words using object detection, and then recognizes characters and words by embedding their crops and retrieving their nearest neighbor from an offline index of exemplar embeddings created from a digital font.

**Localization:** EffOCR supports two widely used backends for localization inference: MMDetection (Chen et al., 2019), which includes state-of-the-art object detection models, and Yolo (Ulyalytics, 2023), which includes fast, efficient object detection models. Users can deploy line, word, and character models from the pre-trained model zoo, that use Yolo v8 (Ulyalytics, 2023) (optimized for efficiency), Yolo v5 (Jocher, 2020) (fewer dependencies) or Cascade R-CNN (Cai and Vasconcelos, 2018) (optimized for accuracy). Pre-trained

<sup>4</sup>TrOCR has a small model (62M parameters), but Carlson et al. (2023) find it is outperformed by the 334M parameter base model by a wide margin on historical documents.

localization models are available for alphabetic English/Latin, Polytonic Greek, and CJK characters (which vary significantly in their aspect ratios and groupings).

**Recognition:** EffOCR recognizes word and character crops using contrastively trained image retrieval models. The EffOCR model zoo currently contains 30 pre-trained models, covering English, Polytonic Greek, and horizontally and vertically-written Japanese. We chose these languages to examine the utility of EffOCR in a high resource setting, in a setting where existing solutions fail, and in an intermediate case.

The EffOCR pre-trained models use a variety of backbones: two lightweight convolutional backbones that are very efficient to deploy (Howard et al., 2019; Maaz et al., 2022), a state-of-the-art CNN encoder (Liu et al., 2022), and three vision transformers (Ali et al., 2021; Li et al., 2022; Liu et al., 2021). For English, there is a word level model that defaults to character recognition when the word is below a default (tunable) cosine similarity threshold, as well as a character-only model.

The documentation provides more guidance on model selection. A description of the training dataset is provided alongside with the trained models such that users can quickly identify the most suitable models for their tasks.

EffOCR can be used off-the-shelf with just a few lines of code:

```
1 import effocr
2 engine = effocr.EffOCR(
3     line_detector = "./line_model",
4     localizer = "./localizer",
5     char_recognizer = "./char_recognizer",
6     word_recognizer = "./word_recognizer"
7 )
8 results = engine.infer('image.jpg')
```

ONNX (ONNX, 2021) integration is an important component, as it allows for efficient CPU deployment and interoperability between deep learning frameworks. All EffOCR stages can optionally employ ONNX-format models and ONNX-runtime inference, and models can be converted to ONNX format within the package. ONNX-runtime increases CPU throughput by up to four times (Jocher, 2020) for YOLO models used in EffOCR, which allows for cost-effective cloud deployment for processing large document sets. ONNX compatibility allows additional model speedups through graph optimizations, quantization, and pruning.

### 3.2 Customized Model Training

Many low-resource settings are poorly served by existing OCR engines, and a central aim of EffOCR is to democratize OCR for these settings by providing a simple interface for custom model training that can be used by researchers and others who have limited experience with deep learning frameworks. Custom training can be initialized using a Yolo object detection model for localization and any timm image encoder model for recognition. In the near future, support for training localization models with MMDetection will be added. This futureproofs EffOCR, as new models are developed.

EffOCR supports logging of a training run on Weights and Biases (Biewald, 2020). It takes industry standard coco json labels as inputs, and hence is compatible with the outputs of a range of both open-source and proprietary labeling softwares. It also exports output in the same format, so that users can easily correct model predictions if desired to speed up labeling.

Model training with EffOCR is highly efficient, *e.g.*, the convolutional backbones can be trained on Google Colab. We trained all models on either a single Nvidia RTX 3090 or A6000 card.

### 3.3 Visualization, Storage and Export

EffOCR comes with a tool to visualize the OCR, side-by-side with the original image, as well as to visualize the line, word, and character predictions. These greatly facilitate quality checking the output and troubleshooting potential problems.

EffOCR offers users different options for data export. The default outputs of EffOCR include line coordinates, word coordinates, character coordinates, and the text associated with each of these annotations. The text for the full image is also assembled in the correct order. Users may choose to export only the assembled text, only text annotations associated with a given level of bounding box (line, word, or character), or all of the above.

### 3.4 User Contributions

By making OCR sample efficient and easy to train, EffOCR aims to promote the reusability and reproducibility of OCR pipelines. This is particularly important for low resource settings and languages, where there is little commercial incentive for product development and few alternatives to crowd-sourcing models. EffOCR users can upload their self-trained models to the EffOCR Hugging

Face hub. Whenever a model is saved, a model card is automatically generated that follows best practices outlined in Hugging Face’s Model Card Guidebook.<sup>5</sup> Moreover, the automatically generated card contains instructions on how to use the model in the context of EffOCR and model-specific architecture and training details in the interest of reproducibility.

### 3.5 Integration with Layout Parser

OCR engines typically detect lines, versus detecting and classifying different layout objects in a document. Many documents have complex layouts - *e.g.*, newspapers have headlines, articles, captions, ads, and headers arranged in complex multicolumn layouts, and tables likewise have different types of information arranged in oftentimes complex layouts. These structures necessitate applying object detection models for document layout analysis, which have been trained to detect the coordinates of each layout object and classify its type (*e.g.*, headline, articles, etc).

To facilitate combining EffOCR with deep learning-based document layout models, wrappers will be integrated into a popular open-source layout detection package, Layout Parser (Shen et al., 2020), that will allow Layout Parser users to call any EffOCR model. Layout Parser also has wrappers to call GCV and Tesseract, which will allow users to easily compare EffOCR output to these other packages to decide what best meets their accuracy and cost objectives. Layout Parser and EffOCR were designed by the same lab, facilitating long-run coordination between the packages.

## 4 Applications

**Scalability:** We have tested the utility of EffOCR with various real-world applications. In the first application, we cheaply and accurately digitized 20 million newspaper page scans from Library of Congress’s Chronicling America collection (Library of Congress, 2022). The resulting dataset, American Stories, is available for download on Hugging Face.<sup>6</sup> Figure 1 illustrates why this is a challenging task: newspapers are extremely heterogeneous in their fonts and image quality. Dell et al. (2023) provide a detailed analysis of the quality of the resulting text dataset.

<sup>5</sup><https://huggingface.co/docs/hub/model-card-guidebook>

<sup>6</sup><https://huggingface.co/datasets/dell-research-harvard/AmericanStories>

|   |   |
|---|---|
| WASHINGTON, April 1—Ambas-                            | WASHINGTON, April 1 Amba-                             |
| FORT WORTH JITNEYS QUIT                               | FORT WORTH JITNEYS QUIT                               |
| General Plan 5-4-31                                   | General Plan 5-4-31                                   |
| State of Tennessee,                                   | State of Tennessee                                    |
| A non-Federal project to furnish free home assistance | A non-Federal project to furnish free home assistance |
| SEED DISTRIBUTION                                     | SEED DISTRIBUTION                                     |
| Iron, Steel and Tin Workers                           | Iron, Steel and Tin Workers                           |
| ADVERSE REPORTS ON DEMENTS NOMINATION.                | ADVERSE REPORTS ON DEMENTS NOMINATION                 |
| IMPROVEMENT IS SHOWN                                  | IMPROVEMENT IS SHOWN                                  |

Figure 1: This figure shows a diversity of examples processed with EffOCR, with predicted transcriptions on the right.

We first trained character EffOCR using synthetic data plus a labeled set of 291 newspaper lines (Carlson et al., 2023), created in a couple of hours. We then bootstrapped word level annotations by creating them with the character level EffOCR model, filtering out lines with a high non-word rate.

With EffOCR, combined with layout analysis using Layout Parser, we could digitize the dataset with a \$60K USD cloud compute budget (plus pipeline development costs). GCV makes significant layout errors when fed full newspaper scans and achieves best performance when fed individual lines. At current prices, digitizing the collection at the line level, since GCV charges per image, would have cost over \$23 million USD. TrOCR Base, the most accurate open-source OCR, would have exceeded our budget by a factor of nearly 50.

**Zero-Shot Performance:** Second, we show that our English lightweight word-level model has strong zero-shot performance on randomly selected document collections from the U.S. National Archives. This model saw only newspapers in training, to test true zero shot performance. We selected a single textline from each of 300 random documents from separate National Archive record groups. EffOCR achieved a 11.2% CER on the diverse collection, compared with a 11.8% CER from Tesseract (Best), a 12.1% CER from EasyOCR, and a 51% CER from TrOCR (Small), which appeared to struggle with blurry and partially obscured text. We suspect the results could be significantly improved by including a random sample of documents from the National Archives in training, to broaden the set of real world documents that the model is exposed to.

All open-source models performed significantly worse than GCV (1.2% CER), but as discussed

earlier cost concerns presently preclude its use at scale. Despite being engineered for low-resource, few-shot contexts, EffOCR remains competitive in high-resource, zero-shot situations.

**Low Resource Settings:** Finally, we use EffOCR to digitize historical Japanese firm level records for vertically written Japanese documents (Teikoku Koshinjo, 1957), where the best available solution (from Baidu OCR) mispredicts over half of characters. We use the evaluation set in Carlson et al. (2023), which consists of randomly selected segments that were double labeled.

Using a training set of 898 labeled table cells, we achieve a CER of 0.7%, 80 times more accurate than the best existing solution. As a result, we are able to study a variety of questions about Japan’s remarkable growth performance that would have been impossible to examine without EffOCR.

To further examine the limits of sample efficiency, we calculate the character classification error when the (character) model only sees one (or up to 5) labeled character(s) for each of the characters that appear in the training set, which comprise 77% of the characters in the test set. This results in character classification errors of 13.4% and 2.0% respectively. While the model does clearly benefit from seeing multiple crops of characters that appear frequently, this illustrates viable few shot performance.

## 5 Limitations

If large portions of a document are illegible, vision-only OCR will not be suitable and language understanding may be helpful for inferring content. For high resource languages such as English when cost is not a concern, users may get the best mileage

from a leading commercial product such as GCV.

Currently, the EffOCR model zoo has pre-trained models supporting typewritten English, Japanese, and Polytonic Greek. Over the coming months, we will be crowd-sourcing annotations (including handwriting) from package users and colleagues. We will use them, along with digital fonts, to pre-train additional models. In addition, users are encouraged to contribute their models.

EffOCR does not currently support handwriting. We started with typewritten documents because there are billions of public domain typeface documents that are of considerable interest to researchers and the general public. We are planning to expand the model zoo to include handwriting and users have already offered to contribute annotations. Synthetic handwriting generators, *e.g.* [Bhunja et al. \(2021\)](#), can provide extensive data for pre-training for scripts that they support, analogous to the use of digital fonts for typeface documents. We will make synthetic handwriting datasets available so that package users can also use them for training their own custom models.

## References

- Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. 2021. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34.
- Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. 2021. Handwriting transformers. *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1086–1094.
- Lukas Biewald. 2020. [Experiment tracking with weights and biases](#). Software available from wandb.com.
- Zhaowei Cai and Nuno Vasconcelos. 2018. Cascade r-cnn: Delving into high quality object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.
- Jacob Carlson, Tom Bryan, and Melissa Dell. 2023. Efficient ocr for building a diverse digital history. *arXiv preprint arXiv:2304.02737*.
- Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. 2019. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.
- Melissa Dell, Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D’Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring. 2023. American stories: A large-scale structured text dataset of historical u.s. newspapers.
- Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.
- Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324.
- Jaidev AI. 2021. Easyocr. <https://github.com/JaidevAI/EasyOCR>.
- Glenn Jocher. 2020. [YOLOv5 by Ultralytics](#).
- Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv preprint arXiv:2004.09095*.
- Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*.
- Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. 2022. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*.
- Library of Congress. 2022. [Chronicling America: Historic American Newspapers](#).
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.
- Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986.
- Lijun Lyu, Maria Koutraki, Martin Krickl, and Besnik Fetahu. 2021. [Neural ocr post-hoc correction of historical corpora](#). *Transactions of the Association for Computational Linguistics*, 9:479–483.
- Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. 2022. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *European Conference on Computer Vision*, pages 3–20. Springer.

- Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. 2021. [Survey of post-ocr processing approaches](#). *ACM Comput. Surv.*, 54(6).
- John Mark Ockerbloom. 2019. [Newspaper copyrights, notices, and renewals](#).
- ONNX. 2021. Onnx runtime. <https://www.onnxruntime.ai>. Version: x.y.z.
- Zejiang Shen, Kaixuan Zhang, and Melissa Dell. 2020. A large dataset of historical japanese documents with complex layouts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 548–549.
- Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.
- Teikoku Koshinjo. 1957. *Teikoku Ginko Kaisha Yoroku*. Teikoku Koshinjo.
- Ultralytics. 2023. Yolo v8 github repository. <https://github.com/ultralytics/ultralytics>.
- Daniel van Strien., Kaspar Beelen., Mariona Coll Ardanuy., Kasra Hosseini., Barbara McGillivray., and Giovanni Colavizza. 2020. [Assessing the impact of ocr quality on downstream nlp tasks](#). In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1: ARTIDIGH*, pages 484–496. INSTICC, SciTePress.
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. <https://github.com/facebookresearch/detectron2>.

## Supplementary Materials

### S-1 Model Architecture and Model Zoo

Figure S-1 shows the EffOCR model architecture, and Table S-1 summarizes the models in the EffOCR model zoo. Readers seeking technical details for the EffOCR models contained in the pre-trained model zoo are referred to the detailed supplementary materials in [Carlson et al. \(2023\)](#).

### S-2 Sample Efficiency

To examine how efficiently EffOCR learns in comparison to leading open source architectures, we train different OCR models from scratch using varying amounts of annotated data. EffOCR-C (Base) is compared to SVTR (implemented via PaddleOCR) ([Du et al., 2022](#)), CRNN (implemented via EasyOCR) ([Shi et al., 2016](#)), and TrOCR ([Li et al., 2021b](#)). All architectures are pre-trained from scratch on 8,000 synthetic text lines, starting from pre-trained checkpoints not customized for OCR when supported by the framework. They are then fine-tuned on the study’s benchmark datasets, with varying train-test-validation splits: 70%-15%-15%, 50%-25%-25%, 20%-40%-40%, 5%-47.5%-47.5%, and 0%-50%-50% (i.e., zero-shot). These exercises are performed for the English newspaper character level models and horizontal Japanese, as vertical Japanese is not supported by the comparison architectures.

Figure S-2 plots the percentage of the benchmark dataset used in training on the x-axis and the CER on the y-axis. On just 99 labeled table cells for Japanese and 21 labeled rows for LoCCA (the 5% train split), EffOCR’s CER is only 5% (Japanese) and 7% (English), showing viable few shot performance. The other architectures remain unusable. EffOCR performs nearly as well using 20% or training data as using 70%, where it continues to outperform all other alternatives. This illustrates that its parsimonious architecture learns efficiently.

### S-3 Training Config Details

The EffOCR package exposes a wide variety of training options and hyperparameters to users. A few key elements are described here, readers looking for more details are directed to the package documentation.

#### *Recognizer Training Options:*

- `timm_model_name` Model name from `timm` ([Wightman, 2019](#)) package used as a base encoder for the recognizer.
- `render_dict` Folder to store crop renders and gold training data locally.
- `font_dir_path` Local path to draw ttf (font) files from, which are used to create character/word renders.
- `hns_txt_path` Local file path to draw hard negative samples from. Hard negative text files are created by default at the end of recognizer training. Most recognizer training applications use two stages, an initial run and a hard negative sampling run.
- `latin_suggested_args` Uses default arguments for alphabetic writing systems such as Latin, Greek, and Cyrillic.

In addition to these options, a wide variety of standard model training parameters are exposed, including learning rate, optimizer options, weight decay, batch size, device selection, and number of training epochs.

#### *Localizer Training Options:*

- `vertical` Whether model should expect characters aligned horizontally (as in English and many Latin scripts) or vertically (as in many character-based scripts).
- `no_words` Detect only characters, not words. Recommended for languages without word groupings.
- `iou_thresh` Training and validation IOU threshold for character/word detection.

- `conf_thresh` Training and validation confidence threshold for character/word detection.

As with the recognizer, other standard training parameters are exposed. In particular, adjusting the image input shape may be valuable for particularly long or short lines.

Hyperparameters and training procedure used to generate models listed in the Model Zoo (Table S-1) are listed in [Carlson et al. \(2023\)](#).

#### **S-4 Visualization**

Figure S-3 shows the EffOCR visualization interface.

#### **S-5 American Stories**

Figure S-4 plots the number of articles in the American Stories dataset, created with EffOCR, across time.

#### **S-6 The Public Domain**

Table S-2 provides detailed information about the requirements for information published in the United States to be in the public domain, in order to give readers a better sense of these collections.

#### **S-7 Inference Speed**

EffOCR implements two features designed to increase computational efficiency. First, both localization and recognition inference is run in a multithreaded fashion, ensuring that compute resources are fully utilized. Second, EffOCR provides support for ONNX runtime and ONNX-format models, which provide up to a 3x speedup on a CPU compared to native PyTorch runtime ([ONNX, 2021](#)). GPUs are typically cost prohibitive for digitization at scale.

Table S-3 provides a comparison between EffOCR and other commonly used OCR frameworks' python implementations. It is important to note that these numbers - across softwares - can vary significantly depending on the hardware resources available. All comparisons are made on four 2200 MHz CPU cores, selected to represent a plausible and relatively affordable research compute setup. EffOCR performance is competitive with other widely used frameworks, with EffOCR (Small) having the fastest performance. Tesseract ([Ooms, 2023](#)) testing used the `pytesseract` package with default settings. EasyOCR ([JaidedAI, 2021](#)) testing used the `easyocr` package with default English settings. PaddleOCR ([PaddlePaddle, 2022](#)) testing used the `paddleocr` package with `use_angle_cls` option and default English settings. TrOCR ([Li et al., 2021a](#)) testing used the `transformers` package implementation, with `trocr-base-printed` and `trocr-small-printed` models for Base and Small tests, respectively. EffOCR testing used default settings with pretrained ONNX English newspaper models from the model zoo.



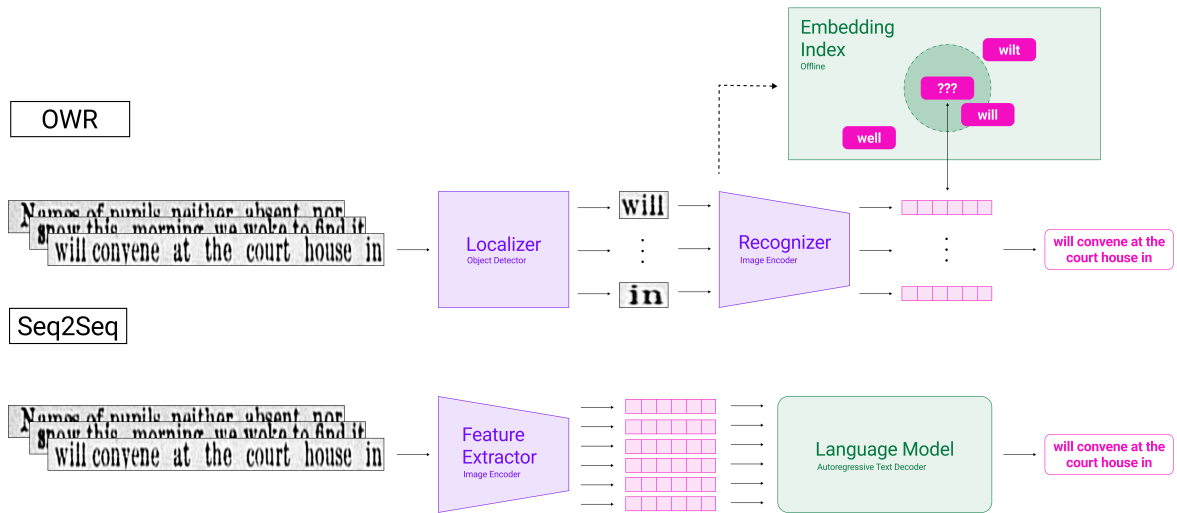


Figure S-1: **EffOCR and Seq2Seq Model Architectures.** This figure represents the EffOCR architecture, as compared to a typical sequence-to-sequence OCR architecture.

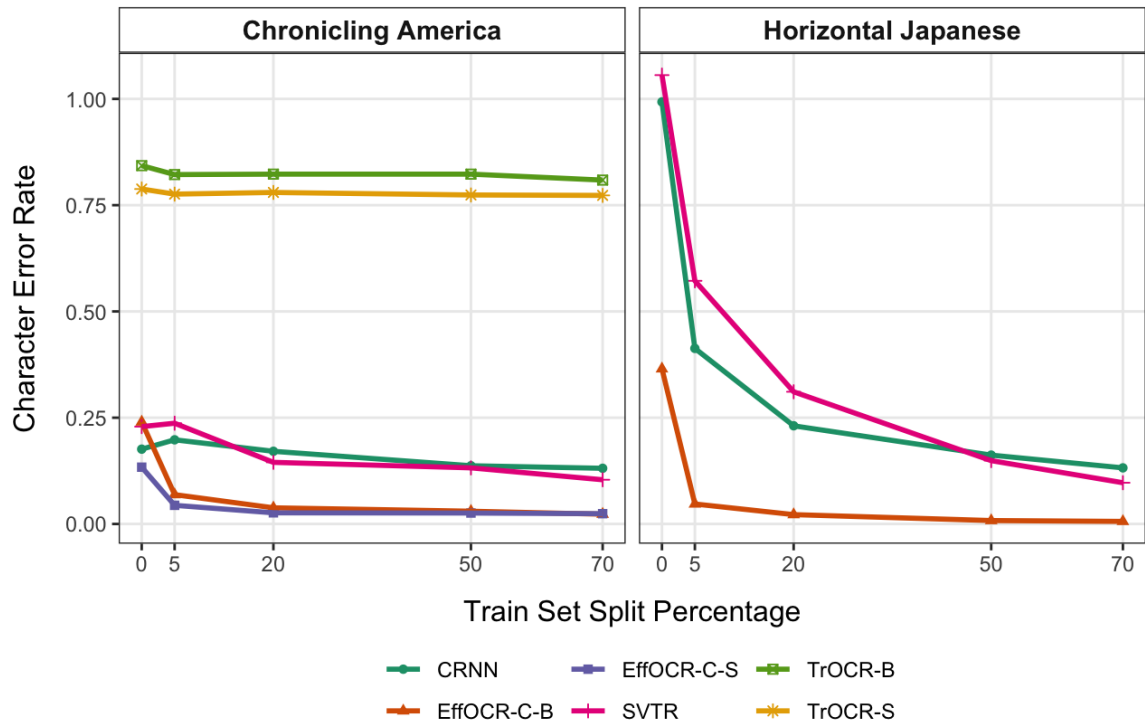


Figure S-2: **Sample Efficiency.** This figure plots the percentage of the benchmark dataset used in training against the character error rate, for different OCR model architectures: CRNN (EasyOCR), SVTR (PaddleOCR), TrOCR (Transformer OCR), and EffOCR small and base convolutional models.

## Original Image

The tug boat Alice, with Captain Rollie Davis and Harry Raymond on board, returned to Juneau early this morning. The Alice left here last week for Thomas Bay to pick up a tow of piles for the Treadwell company. The tow, which consists of about 400 piles, is tied up at Taku Harbor as the high winds prevented towing.

"We bucked bucked the wind down and we bucked the wind back, part of the time making less than a half mile an hour," said Harry Raymond this morning.

As soon as the weather moderates the Alice will leave and bring in the tow and will then return to Thomas Bay and pick up a tow of piles for delivery in Juneau.

## Transcribed Text

The tug boat Alice. with Captain Rollie Davis and Harry Raymond on board. returned to Juneau early this morning. The Alice left here last week for Thomas Bay to pick up a tow Of piles for the Treadwell company. The tow, which consists OF about 400 piles. is tied up at Taku Harbor as the high winds prevented towing.?

I'VE bucked bucked the wind down and we bucked the wind back. part of the time making less than n half mile on hour" said Harry Raymond this morning.?

AS soon as the weather moderates the Alice will leave and bring in the Tow and will then return to Thomas Bay and pick up a tow of piles for delivery in Juneau.

## Line Detections

The tug boat Alice, with Captain Rollie Davis and Harry Raymond on board, returned to Juneau early this morning. The Alice left here last week for Thomas Bay to pick up a tow of piles for the Treadwell company. The tow, which consists of about 400 piles, is tied up at Taku Harbor as the high winds prevented towing.

"We bucked bucked the wind down and we bucked the wind back, part of the time making less than a half mile an hour," said Harry Raymond this morning.

As soon as the weather moderates the Alice will leave and bring in the tow and will then return to Thomas Bay and pick up a tow of piles for delivery in Juneau.

## Word Detections

The tug boat Alice, with Captain Rollie Davis and Harry Raymond on board, returned to Juneau early this morning. The Alice left here last week for Thomas Bay to pick up a tow of piles for the Treadwell company. The tow, which consists of about 400 piles, is tied up at Taku Harbor as the high winds prevented towing.

"We bucked bucked the wind down and we bucked the wind back, part of the time making less than a half mile an hour," said Harry Raymond this morning.

As soon as the weather moderates the Alice will leave and bring in the tow and will then return to Thomas Bay and pick up a tow of piles for delivery in Juneau.

## Character Detections

The tug boat Alice, with Captain Rollie Davis and Harry Raymond on board, returned to Juneau early this morning. The Alice left here last week for Thomas Bay to pick up a tow of piles for the Treadwell company. The tow, which consists of about 400 piles, is tied up at Taku Harbor as the high winds prevented towing.

"We bucked bucked the wind down and we bucked the wind back, part of the time making less than a half mile an hour," said Harry Raymond this morning.

As soon as the weather moderates the Alice will leave and bring in the tow and will then return to Thomas Bay and pick up a tow of piles for delivery in Juneau.

Figure S-3: **Visualization.** This figure shows the EffOCR visualization interface.

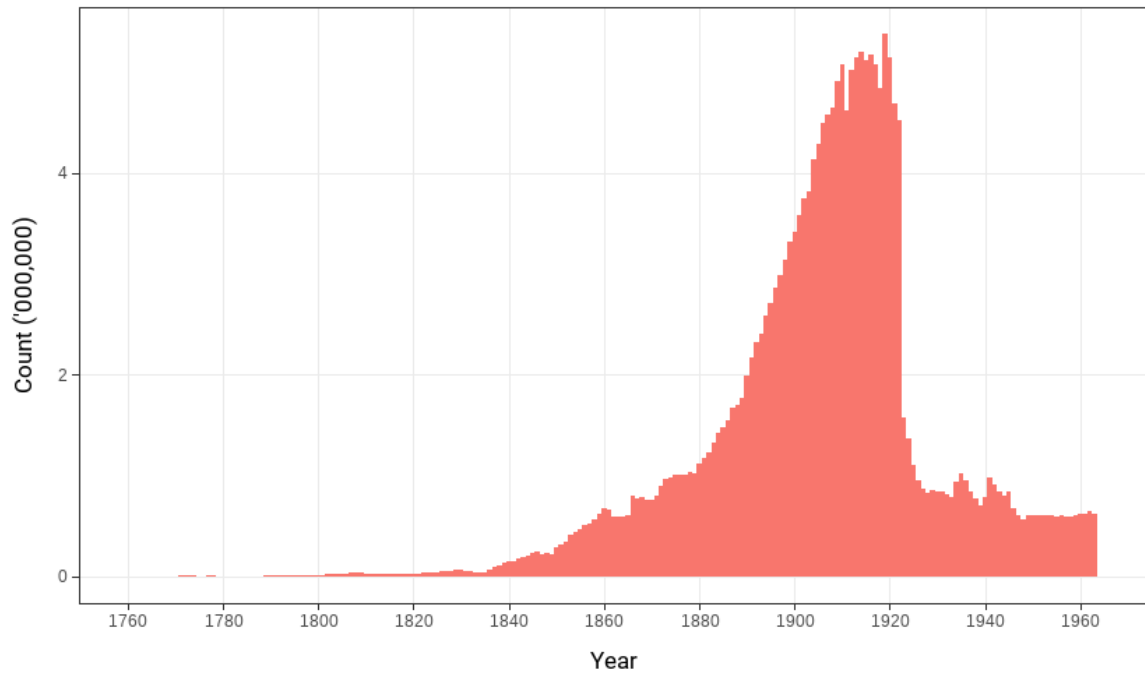


Figure S-4: **American Stories**. This figure plots the number of articles in the American Stories dataset, created with EffOCR, across time.

| Training Set           | Line Detection | Localizer |          | Word Recognition |          | Character Recognition |          |     |          |      |
|------------------------|----------------|-----------|----------|------------------|----------|-----------------------|----------|-----|----------|------|
|                        | YOLO           | YOLO      | MaskRCNN | MobileNetV3      | EdgeNeXt | MobileNetV3           | EdgeNeXt | ViT | ConvNeXt | XCiT |
| English Newspapers     | ✓              | ✓         | ✓        | ✓                | ✓        | ✓                     | ✓        | ✓   | ✓        | ✓    |
| English Mixed Archival | ✓              | -         | ✓        | ✓                | -        | ✓                     | -        | -   | -        | -    |
| Japanese Vertical      | ✓              | ✓         | ✓        | N/A              | N/A      | ✓                     | ✓        | ✓   | ✓        | ✓    |
| Japanese Horizontal    | ✓              | ✓         | ✓        | N/A              | N/A      | ✓                     | ✓        | ✓   | ✓        | ✓    |
| Polytonic Greek        | ✓              | ✓         | -        | N/A              | N/A      | ✓                     | -        | -   | ✓        | -    |

Table S-1: Models Currently Available in the EffOCR Model Zoo. Note Japanese models do not use word-level recognition.

| Date of Publication            | Conditions   | Copyright Term   |
|--------------------------------|--|--|
| <b><i>Public Domain</i></b>    |  |  |
| <b>Anytime</b>                 | <b>Works prepared by an officer/employee of the U.S. Government as part of their official duties</b> | <b>None</b>  |
| <b>Before 1928</b>             | <b>None</b>  | <b>None. Copyright expired.</b>  |
| <b>1928 through 1977</b>       | <b>Published without a copyright notice</b>  | <b>None. Failure to comply with required formalities</b>   |
| <b>1978 to 1 March 1989</b>    | <b>Published without notice and without subsequent registration within 5 years</b>                   | <b>None. Failure to comply with required formalities</b>   |
| <b>1928 through 1963</b>       | <b>Published with notice but copyright was not renewed</b>   | <b>None. Copyright expired</b>   |
| <b><i>Copyrighted</i></b>      |  |  |
| 1978 to 1 March 1989           | Published without notice, but with subsequent registration within 5 years                            | 70 (95) years after the death of author (corporate author)   |
| 1928 through 1963              | Published with notice and the copyright was renewed  | 95 years after publication   |
| 1964 through 1977              | Published with notice  | 95 years after publication   |
| 1978 to 1 March 1989           | Created after 1977 and published with notice   | 70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier |
| 1978 to 1 March 1989           | Created before 1978 and first published with notice in the specified period                          | The greater of the term specified in the previous entry or 31 December 2047                        |
| From 1 March 1989 through 2002 | Created after 1977   | 70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier |
| From 1 March 1989 through 2002 | Created before 1978 and first published in this period   | The greater of the term specified in the previous entry or 31 December 2047                        |
| After 2002                     | None   | 70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier |

Table S-2: This table summarizes U.S. copyright law, based on a similar table produced by the Cornell libraries. For concision, we focus on works initially published in the United States. A variety of other cases are also covered at <https://guides.library.cornell.edu/copyright>.

| Model         | Textline/s | Article/s |
|---------------|------------|-----------|
| EffOCR Base   | 0.46       | 0.02      |
| EffOCR Small  | 21.07      | 1.08      |
| Tesseract     | 4.47       | 0.21      |
| EasyOCR       | 19.80      | 1.03      |
| PaddleOCR     | 13.56      | 0.61      |
| TrOCR (Base)  | 0.43       | 0.02      |
| TrOCR (Small) | 0.97       | 0.05      |

Table S-3: Comparison of EffOCR speeds with other popular OCR frameworks in CPU environment. Tests included both Textline (single lines of text) and Article (5-40 lines of text) examples.

## References

- Jacob Carlson, Tom Bryan, and Melissa Dell. 2023. Efficient ocr for building a diverse digital history. *arXiv preprint arXiv:2304.02737*.
- Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.
- JaidevAI. 2021. Easyocr. <https://github.com/JaidevAI/EasyOCR>.
- Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021a. Trocr github repository. <https://github.com/microsoft/unilm/tree/master/trocr>.
- Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021b. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*.
- ONNX. 2021. Onnx runtime. <https://www.onnxruntime.ai>. Version: x.y.z.
- J Ooms. 2023. Tesseract: Open source ocr engine.
- PaddlePaddle. 2022. PaddleOCR.
- Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.
- Ross Wightman. 2019. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>.



# Author Index

- Ahmadi, Seyedeh Fatemeh, 274  
Ahmed, Shafiuddin Rehan, 381  
Akbik, Alan, 1  
Akiki, Christopher, 140  
Amar, Shmuel, 554  
Arkhipkin, Vladimir, 286  
Arora, Abhishek, 579  
Ashraf Vaghefi, Saeid, 21  
Authur, Russell, 495
- Balasubramanian, Niranjan, 328  
Baldwin, Timothy, 446  
Bar-Haim, Roy, 403  
Barbour, Tanner, 373  
Belyy, Anton, 124  
Bertsch, Amanda, 413  
Bhattacharjee, Bishwaranjan, 229  
Bian, Jiang, 246  
Bing, Lidong, 543  
Bingler, Julia, 21  
Bonn, Julia, 381  
Boreshban, Yasaman, 274  
Borse, Santosh, 229  
Bransom, Erin, 495  
Brown, Susan Windisch, 365  
Bryan, Tom, 579  
Burke, Martin, 389
- Cai, Jon, 381  
Candel, Arno, 82  
Candra, Stefan, 495  
Cao, Pengfei, 479  
Carlson, Jacob, 579  
Carvalhais, Nuno, 70  
Cattan, Arie, 403  
Chambers, Nathanael, 328  
Chandrasekhar, Yoganand, 495  
Chang, Joseph Chee, 495  
Chang, Kevin Chen-Chuan, 462  
Chang, Yongzhu, 99  
Chao, WenHan, 177  
Chen, He, 566  
Chen, Junhao, 479  
Chen, Keyu, 527  
Chen, Qihang, 99  
Chen, Xiang Anthony, 161  
Chen, Yingda, 566  
Chen, Yubo, 479
- Chen, Zhi, 186  
Cheng, Chen, 566  
Chiu, Yu Cheung, 167  
Chiu, Yu Ying, 167  
Cohen, Jonathan, 431  
Colesanti-Senni, Chiara, 21  
Conde, Marcos, 82  
Constantin, Stefan, 12
- Dagan, Ido, 403, 554  
Deguchi, Jun, 52  
Dell, Melissa, 579  
Dernoncourt, Franck, 318  
Devare, Sugam, 328  
Dimitrov, Denis, 286  
Ding, Rui, 346  
Ding, Yixi, 518  
Ding, Ziqi, 422  
Dinh, Tu Anh, 12  
Dinu, Razvan, 431  
Dou, Yao, 336  
Downey, Doug, 403, 495  
Du, Pengfan, 479  
Durrett, Greg, 328
- Eden, Lilach, 403  
Engländer, Leon, 149  
Erk, Katrin, 328
- Fadeeva, Ekaterina, 446  
Fatahi Bayat, Farima, 124  
Fedyanin, Kirill, 446  
Feng, Shutong, 106  
Ferraro, Francis, 328
- Gao, Jianfeng, 106  
Gasic, Milica, 106  
Geishauser, Christian, 106  
Ghassem-Sani, Gholamreza, 274  
Ghosh, Sayontan, 328  
Glavaš, Goran, 296  
Golde, Jonas, 1  
Goncharova, Elizaveta, 446  
Gostlow, Glen, 21  
Gu, Tianle, 527  
Gunapati, Gautham, 328  
Guo, Honglin, 527  
Guo, Qipeng, 527

Gurevych, Iryna, 149  
 Haffari, Gholamreza, 90  
 Hahn, Udo, 218  
 Hajialigol, Daniel, 373  
 Haller, Patrick, 1  
 Hamborg, Felix, 1  
 Han, Benjamin, 124  
 Han, Jiawei, 365, 389  
 Han, Shi, 346  
 Han, Zeyu, 237  
 He, Tianyu, 246  
 He, Xuanli, 90  
 Hearst, Marti A., 495  
 Heck, Michael, 106  
 Heineman, David, 336  
 Hong, Jiawei, 527  
 Hope, Tom, 403  
 Hoshi, Yasuto, 52  
 Hou, Lei, 508  
 Hraška, Peter, 471  
 Hu, Vivian, 389  
 Hu, Zi-Yuan, 186  
 Huang, Fei, 566  
 Huang, Jie, 462  
 Huang, Minlie, 106  
 Huang, Shijia, 186  
 Huang, Yongfeng, 186  
 Huber, Christian, 12  
 Huff, Regan, 495  
  
 Iana, Andreea, 296  
 Ilyas, Ihab, 124  
 Imhof, Timo, 149  
  
 Jeblick, Maximilian, 82  
 Ji, Heng, 365, 389  
 Jiang, Lin, 99  
 Jiang, Minhao, 389  
 Jiao, Yizhu, 389  
  
 Kaknes, Derek, 373  
 Kamani, Fatemeh, 274  
 Kan, Min-Yen, 264, 518  
 Kantor, Yoav, 403  
 Kargupta, Priyanka, 389  
 Khorshidi, Samira, 124  
 Koneru, Sai, 12  
 Koupaee, Mahnaz, 328  
 Kraus, Mathias, 21  
 Kuehl, Bailey, 495  
  
 Kuts, Angelina, 286  
 Kuznetsov, Andrey, 286  
  
 Laban, Philippe, 161  
 Lai, Viet, 318  
 Lal, Yash Kumar, 328  
 Lawonn, Kai, 70  
 Lee, Chun Ming, 82  
 Leippold, Markus, 21  
 Li, Chenliang, 566  
 Li, Hongxiang, 389  
 Li, Juanzi, 508  
 Li, Sha, 365  
 Li, Xin, 543  
 Li, Yanyang, 186  
 Li, Yuhan, 237  
 Li, Yunyao, 124  
 Li, Zhaolin, 12  
 Liem, David, 373  
 Lin, Dahua, 186  
 Lin, Hsien-chin, 106  
 Lin, Jimmy, 140  
 Liu, Danni, 12  
 Liu, Kang, 479  
 Liu, Qian, 518  
 Liu, Shengping, 479  
 Liu, Tengxiao, 527  
 Liu, Xiaoran, 527  
 Liu, Xuan, 389  
 Liu, Xukun, 237  
 Liu, Yuchen, 237  
 Lo, Kyle, 495  
 Loftsson, Hrafn, 256  
 Lohr, Christina, 218  
 Lu, Peiling, 246  
 Lu, Xinyuan, 264  
 Lubis, Nurul, 106  
 Luo, Leo, 389  
 Luo, Zhunchen, 177  
 Lv, Kai, 527  
 Lyu, Michael, 186  
  
 Ma, Pingchuan, 346  
 Maltseva, Anastasia, 286  
 Martin, James H., 381  
 McKinney, Jon, 82  
 Meuschke, Monique, 70  
 Milbauer, Jeremiah, 422  
 Mirbostani, Seyed Morteza, 274  
 Mirroshandel, Seyed Abolghasem, 274  
 Miyashita, Daisuke, 52

Mooney, Raymond, 328  
 Morioka, Yasuhiro, 52  
 Mullov, Carlos, 12  
 Murakhovs'ka, Lidiya, 161  
  
 Nachum, Niv, 554  
 Nakov, Preslav, 264  
 Neubig, Graham, 413  
 Newman, Benjamin, 495  
 Ng, Youyang, 52  
 Ngo, Nghia, 318  
 Nguyen, Chien, 318  
 Nguyen, Khanh Duy, 365  
 Nguyen, Thai Binh, 12  
 Nguyen, Thien, 318  
 Nguyen, Thuat, 318  
 Ni, Jingwei, 21  
 Niehues, Jan, 12  
 North, Chris, 373  
  
 Ogundepo, Odunayo, 140  
 Oladipo, Akintunde, 140  
 Ouyang, Siru, 389  
  
 Palmer, Martha, 365, 381  
 Pan, Liangming, 264  
 Panchenko, Alexander, 286, 446  
 Panov, Maxim, 446  
 Parisien, Christopher, 431  
 Paul, Indraneil, 149  
 Paulheim, Heiko, 296  
 Pavlov, Igor, 286  
 Peng, Baolin, 106  
 Peng, Hao, 508  
 Peng, Zhiyuan, 237  
 Petrakov, Sergey, 446  
 Pfeiffer, Jonas, 149  
 Pfeiffer, Pascal, 82  
 Pham, Ngoc-Quan, 12  
 Piktus, Aleksandra, 140  
 Poth, Clifton, 149  
 Potthast, Martin, 140  
 Pu, Jiashu, 99  
 Purkayastha, Sukannya, 149  
  
 Qian, Kun, 124  
 Qin, Yanxia, 518  
 Qiu, Xipeng, 527  
  
 Razzhigaev, Anton, 286  
 Rebedea, Traian, 431  
  
 Reichstein, Markus, 70  
 Retkowski, Fabian, 12  
 Risch, Julian, 1  
 Rosenthal, Sara, 229  
 Rossi, Ryan, 318  
 Ruder, Sebastian, 149  
 Rush, Alexander, 311  
 Ryabov, Ilya, 286  
  
 Sang, Yisi, 124  
 Sartran, Laurent, 353  
 Schimanski, Tobias, 21  
 Shakhmatov, Arseniy, 286  
 Shapira, Ori, 554  
 Shareghi, Ehsan, 90  
 Shelmanov, Artem, 446  
 Shen, Hua, 237  
 Shen, Weizhou, 566  
 Shen, Yanzhen, 389  
 Shen, Zejiang, 495  
 Shi, Hongzhu, 566  
 Shojaee, Gita, 274  
 Sil, Avirup, 229  
 Singer, Philipp, 82  
 Singh, Amanpreet, 495  
 Skala, Daniel, 471  
 Slobodkin, Aviv, 554  
 Soldaini, Luca, 495  
 Song, Kaitao, 246  
 Sreedhar, Makesh Narsimhan, 431  
 Stambach, Dominik, 21  
 Stanojević, Miloš, 353  
 Steingrimsson, Steinthor, 256  
 Sterz, Hannah, 149  
 Su, Xiaohui, 186  
 Suchocki, Reece, 365  
 Sun, Jimeng, 373  
 Sun, Yu, 527  
 Sučik, Samuel, 471  
 Šuppa, Marek, 471  
 Švec, Andrej, 471  
  
 Tan, Xu, 246  
 Tatsuno, Kento, 52  
 Teng, Zhiyang, 131  
 Tillmann, Christoph, 229  
 Torii, Osamu, 52  
 Trivedi, Aashka, 229  
 Tsvigun, Akim, 446  
  
 Ugan, Enes, 12

Vallurupalli, Sai, 328  
 van Niekerk, Carel, 106  
 Vashurin, Roman, 446  
 Vasilev, Daniil, 446  
 Vazhentsev, Artem, 446  
 Viswanathan, Vijay, 413  
 Voigt, Henrik, 70  
 Vu, Thuy-Trang, 90  
 Vuli, Ivan, 149

Waibel, Alexander, 12  
 Wan, Dazhen, 106  
 Wang, Liwei, 186  
 Wang, Qian, 21  
 Wang, Shuai, 346  
 Wang, Xiaozhi, 508  
 Wang, Xuan, 373, 389  
 Wang, Yile, 131  
 Wang, Zeqiang, 131  
 Wang, Zhilin, 167  
 Wang, Zimu, 508  
 Way, Andy, 256  
 Webersinke, Nicolas, 21  
 Wekhof, Tobias, 21  
 Weld, Daniel, 495  
 Wilhelm, Chris, 495  
 Wright-Bettner, Kristin, 381  
 Wu, Chien-Sheng, 161  
 Wu, Fei, 124  
 Wu, Jiageng, 131  
 Wu, Qi, 177  
 Wu, Tongshuang, 413, 422  
 Wu, Zhijin, 422  
 Wu, Zhikai, 566

Xiao, Jinfeng, 389  
 Xie, Haining, 479  
 Xing, Shuhao, 527  
 Xu, Binfeng, 237  
 Xu, Dongkuan, 237  
 Xu, Haiyang, 566  
 Xu, Wei, 336

Yan, Hang, 527  
 Yan, Ming, 566  
 Yang, Chunxu, 161  
 Yang, Jie, 131  
 Yang, Yuqing, 527  
 Yao, Daphne, 373  
 Yao, Feng, 508  
 Yao, Ziyu, 237  
 Ye, Wei, 246  
 Yu, Dingyao, 246  
 Yu, Tingyu, 21  
 Yue, Murong, 237

Zamarron, Angele, 495  
 Zarrie, Sina, 70  
 Zeng, Kaisheng, 508  
 Zhang, Baoli, 479  
 Zhang, Dongmei, 346  
 Zhang, Hang, 543  
 Zhang, Ji, 566  
 Zhang, Le, 99  
 Zhang, Rong, 229  
 Zhang, Rongsheng, 99  
 Zhang, Shikun, 246  
 Zhang, Shuo, 527  
 Zhang, Xinyu, 140  
 Zhang, Zheng, 106  
 Zhang, Zhicheng, 566  
 Zhang, Zixuan, 365  
 Zhao, Chenyang, 413  
 Zhao, Huimin, 389  
 Zhao, Jianqiao, 186  
 Zhao, Jun, 479  
 Zheng, Duo, 186  
 Zhong, Ming, 389  
 Zhong, Xianrui, 389  
 Zhou, Bobby, 389  
 Zhou, Jingren, 566  
 Zhou, Wenmeng, 566  
 Zhou, Xian, 177  
 Zhu, Chuzhao, 508  
 Zhu, Kerui, 462  
 Zhu, Qi, 106  
 Zhu, Xiaochen, 106